1988

A data analysis system for unsteady turbulence measurements

Johnson, Donald K.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A Data Analysis System for
Unsteady Turbulence Measurements

by

DONALD K. JOHNSON

September 1988

Thesis Co-Advisors: P.N. Ilacqua
R.M. Howard

Approved for public release; distribution
is unlimited.

ECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

a. REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
a. SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (*If applicable*) Code 67	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
:. ADDRESS (*City, State, and ZIP Code*) Monterey, California 93943-5000	7b. ADDRESS (*City, State, and ZIP Code*) Monterey, California 93943-5000
a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
. ADDRESS (*City, State, and ZIP Code*)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.
. TITLE (*Include Security Classification*) A DATA ANALYSIS SYSTEM FOR UNSTEADY TURBULENCE MEASUREMENTS	
. PERSONAL AUTHOR(S) Johnson, Donald K.	
a. TYPE OF REPORT Master's Thesis 13b TIME COVERED FROM _____ TO _____ 14. DATE OF REPORT (*Year, Month, Day*) 1988 September 15 PAGE COUNT 238	
SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.	
COSATI CODES FIELD / GROUP / SUB-GROUP	18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*) Unsteady Turbulence, Non-Stationary Statistical Data Analysis, Random Data, Power Spectral Density

. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

A data analysis system has been developed to analyze unsteady turbulence measurements obtained in the boundary layer of an airfoil subjected to periodic turbulent pulse disturbances such as a propeller slipstream. Specific algorithms for analyzing the non-stationary data are identified, developed and implemented. Where alternate algorithms are developed, each is evaluated and the best method is recommended based on specified criteria. The statistical parameters used to characterize the unsteady turbulent boundary layer include the non-stationary mean velocity, turbulence intensity, power spectral density and the autocorrelation and cross correlation functions. Since the data is periodic, both ensemble averaging and special non-stationary empirical model techniques are employed. Each of the algorithms has been implemented in the FORTRAN language and examples are presented using representative test case data.

). DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
a. NAME OF RESPONSIBLE INDIVIDUAL Richard M. Howard	22b TELEPHONE (*Include Area Code*) 408-646-2870 22c. OFFICE SYMBOL Code 67Ho

D FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

☆ U.S. Government Printing Office: 1986—606-24.

A Data Analysis System for Unsteady Turbulence Measurements

by

Donald K. Johnson
B.S., California State Polytechnic University, Pomona, 1977

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL
September 1988

ABSTRACT

A data analysis system has been developed to analyze unsteady turbulence measurements obtained in the boundary layer of an airfoil subjected to periodic turbulent pulse disturbances such as in a propeller slipstream. Specific algorithms for analyzing the non-stationary data are identified, developed and implemented. Where alternate algorithms are developed, each is evaluated and the best method is recommended based on specified criteria. The statistical parameters used to characterize the unsteady turbulent boundary layer include the non-stationary mean velocity, turbulence intensity, power spectral density and the autocorrelation and cross correlation functions. Since the data are periodic, both ensemble averaging and special non-stationary empirical model techniques are employed. Each of the algorithms has been implemented in the FORTRAN language and examples are presented using representative test case data.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

I wish to thank my thesis advisors, Professor Paul Ilacqua and Professor Rick Howard, for their expert guidance and encouragement during the course of this project. Special thanks also goes to Dr. Hal Fredricksen for his extra effort in reviewing very rough drafts of this document, and for initially approving the thesis topic against his better judgment. I would like to thank Professor Ralph Hippenstiel for his assistance on some of the intricacies of spectral analysis.

I wish to express my appreciation to the management of the Air Force Flight Test Center for giving me the opportunity to attend school full time. I look forward to getting back among the airplanes and blue sky.

Finally, I would like to thank my parents Richard and Patricia Johnson, who made it all possible.

I. INTRODUCTION

The need to understand the nature of turbulent fluid flow has long been a field of aerodynamic research. Much of the productive work in the field has been experimental in nature. Since turbulence is a stochastic or random process, analysis of experimental data requires the use of statistical methods. Due to the complex nature of turbulent fluid flow and its most frequent applications, most of the experimental work has been restricted to steady (statistically stationary) turbulence. Statistical methods for stationary random processes are well understood. Recent investigations by Holmes, Obara and Yip [Ref. 1] and Howard [Ref. 2] have begun to extend experimental research into unsteady (non-stationary) turbulent fluid flow conditions. The most recent work by Howard consists of an investigation of the time varying characteristics of the boundary layer of an airfoil immersed in a propeller slipstream. These experiments require the application of non-stationary statistical data analysis methods.

Future wind tunnel tests have been proposed by Howard [Ref. 2] to continue investigations into the time dependent nature of the boundary layer of an airfoil subjected to periodic turbulent wake disturbances. The proposed

experimental work has motivated the need for non-stationary statistical data analysis methods with a sound mathematical and statistical basis.

This thesis describes an effort to develop a complete data analysis system to support the proposed wind tunnel tests. The data analysis may also be applied to other unsteady turbulent flows, such as internal flows in turbomachinery. Specific algorithms for analyzing non-stationary turbulent boundary layer velocity data are identified, developed and implemented. Where alternate algorithms are identified, each algorithm is evaluated and the best method is recommended based on specified criteria.

Chapter II presents a brief overview of turbulent fluid flow and boundary layer theory. Chapter II includes a description of the specific statistical quantities used to characterize turbulence, a brief description of the results of the recent tests by Howard [Ref. 2] and the detailed objectives and design constraints for the unsteady turbulence data analysis system. Chapter III briefly reviews the basic mathematical and statistical concepts which were applied to develop the data analysis methods. Chapter IV presents the detailed explanations of each algorithm and includes test case examples used to evaluate the operation and suitability of the computer programs. Conclusions and recommendations are presented in Chapter V. Appendix A presents some derivations

and analyses not included in Chapter IV. Appendix B contains the user's guide and FORTRAN source code for each of the computer programs.

II. NATURE OF THE PROBLEM

The purpose of this chapter is to present background material on the topic of turbulent fluid flow and boundary layer theory, to briefly review recent work completed on the behavior of unsteady boundary layers in a propeller slipstream, and then to state the specific objectives of this thesis. The following references were found helpful in the preparation of this chapter: Bradshaw [Ref. 3], Cebeci and Smith [Ref. 4], Frost and Moulden [Ref. 5] and Howard [Ref. 2].

A. WHAT IS TURBULENCE?

The phenomenon of fluid turbulence occurs in nearly all fluid flow situations. To paraphrase the definitions of turbulence presented in Cebeci and Smith [Ref. 4]:

> Turbulent fluid motion is an irregular fluid flow condition which, in general, appears when fluids flow past solid surfaces or when neighboring streams of fluid pass over one another in shear layers. Turbulence is characterized by a random high frequency variation in various quantities (e.g., velocity, pressure, temperature, etc.) with time and space coordinates, so that statistically distinct average values can be discerned.

Turbulent fluid flow is a "mixing" flow where the fluid streamlines are intersecting, twisting and curling about each other. In contrast, laminar flow is characterized by smooth,

parallel streamlines with no mixing and no random fluctuations in the flow parameters.

Bradshaw [Ref. 3] defines the study of turbulence as the art of understanding the Navier-Stokes equations without actually solving them. The Navier-Stokes equations, which have been derived from Newton's second law of motion, are the governing equations of fluid flow and are three dimensional non-linear partial differential equations in four independent variables (three space dimensions and time). These equations express complicated relationships between the fluid quantities and have proven resistant to explicit solution, resulting in "too many unknowns and not enough equations". The primary deterrence to the solution of the Navier-Stokes equations has been the lack of a suitable mathematical turbulence model.

Therefore, much of the productive study of turbulent fluid flow results from statistical analysis of empirical (measured) data to characterize the nature of turbulence. The velocity (or other parameters of interest) of a turbulent fluid in motion is often modeled by defining the instantaneous velocity as a sum of a mean velocity u and a randomly fluctuating velocity, u', with a zero mean, as in the equation:

$$\underset{\text{Instantaneous Velocity}}{u(t)} = \underset{\text{Mean Velocity}}{\bar{u}} + \underset{\text{Fluctuating Velocity}}{u'(t)} \qquad \text{(II-1)}$$

where u(t) is the velocity in the direction of the mean flow (the x direction). The y and z turbulent velocity components

are similarly defined using the variables v and w respectively except that the mean velocities v and w are zero. In the context of statistics, steady turbulence is defined as a fluid flow with a constant mean velocity and a constant variance high frequency fluctuating velocity component of constant variance (statistically stationary). Unsteady turbulence is characterized as a fluid flow with time-dependent mean velocity and/or time-dependent variance fluctuating velocity component (non-stationary). Figure II-1 [from Bradshaw, Ref. 3] presents an illustration of steady vs. unsteady turbulent flows. In contrast to turbulent fluid flow, laminar flow (either steady or unsteady) is defined as a flow in which the instantaneous velocity is essentially equal to the mean velocity (i.e., the fluctuating velocity component is zero or very small: on the order of 0.02% of the mean velocity).

B. BOUNDARY LAYERS

A boundary layer is a thin region of fluid flow next to a solid surface such as an airfoil in which strong viscous effects exist. Due to fluid viscosity, the layer of fluid particles touching the surface are attached to the surface and have no velocity relative to the surface. The fluid velocity increases with distance from the surface until the velocity equals that outside the boundary layer. The way in which the velocity increases across the boundary layer is called the

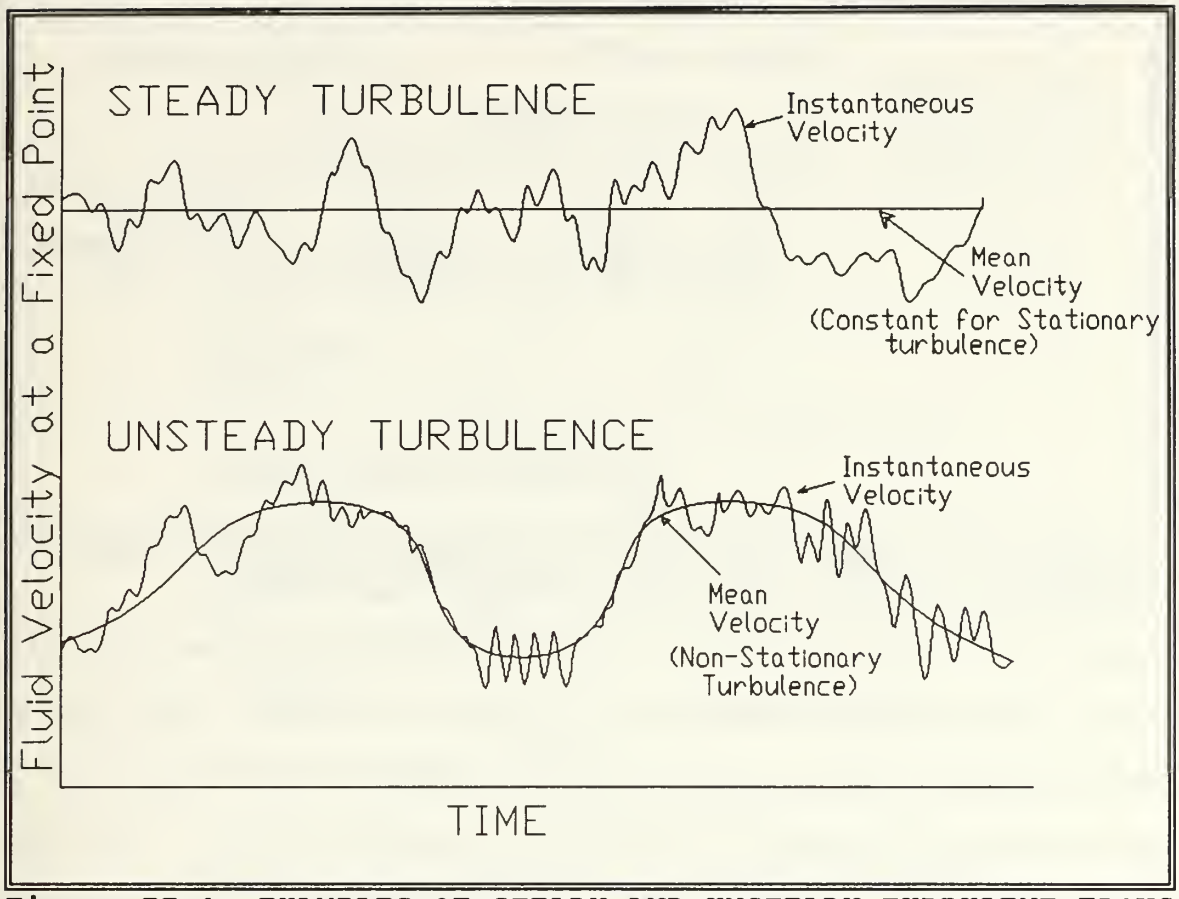

Figure II-1. EXAMPLES OF STEADY AND UNSTEADY TURBULENT FLOWS [BRADSHAW, REF. 3].

boundary layer velocity profile (see Figure II-2). The boundary layer typically grows in thickness as the fluid flows across the surface from the leading edge.

Boundary layer velocity profiles have been divided into different categories based on their shape. The two most important profiles are called laminar and turbulent velocity profiles and each profile has its own characteristic shape. Figure II-3 presents examples of typical laminar and turbulent boundary layer velocity profiles. Due to the mixing and momentum transfer in a turbulent boundary layer, the turbulent

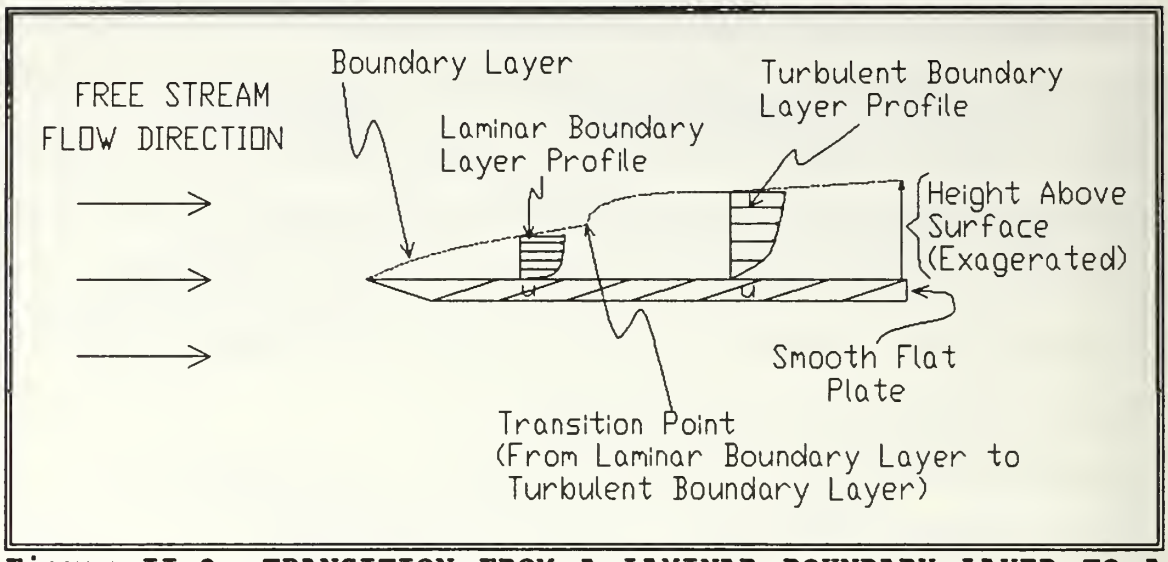

Figure II-2. TRANSITION FROM A LAMINAR BOUNDARY LAYER TO A TURBULENT BOUNDARY LAYER ON A SMOOTH FLAT PLATE.

boundary layer profile is characterized by a higher velocity near the surface compared to a laminar profile. The higher velocity near the surface causes a higher frictional drag on the surface compared to the laminar boundary layer. Boundary layers near the leading edge of smooth surfaces usually start out laminar. Viscous forces cause boundary layers to transition from laminar to turbulent at some distance back from the leading edge. One of the methods of reducing aerodynamic drag is to delay the transition of the boundary layer from laminar to turbulent for as long as possible on the entire surface. The boundary layer can be sensitive to surface roughness or to external disturbances in the flow, such as turbulent wakes from control surfaces or propellers. Such disturbances can cause early transition of the boundary layer from the laminar to the turbulent case.

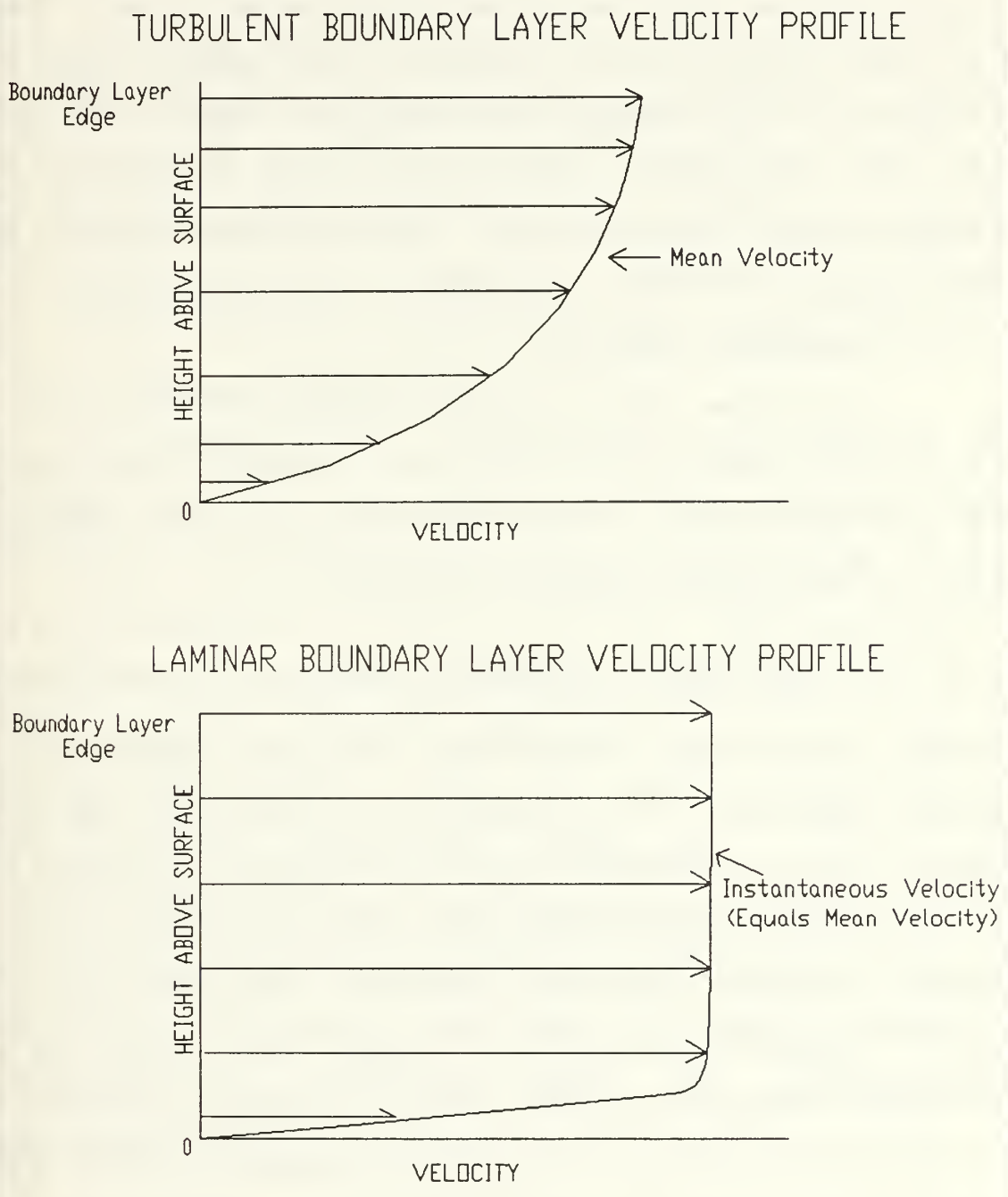

Figure II-3. LAMINAR VERSUS TURBULENT BOUNDARY LAYER PROFILE.

C. PHYSICAL QUANTITIES OF INTEREST

This section presents some of the physical quantities used to describe the characteristics of a turbulent flow. These statistical parameters are defined in this chapter using the terminology of the aerodynamicist such as in Cebeci and Smith [Ref. 4]. Chapter III then reviews the statistics and mathematics of random processes which describe many of the same parameters in the terminology of the statistician.

1. Turbulence Intensity

Turbulence intensity is a measure of the relative energy of the turbulence. Turbulence intensity is defined as the root mean square of the fluctuating velocity component where the mean square value is given by:

$$\overline{(u')^2} = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} (u_i')^2 = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} (u_i-\bar{u})^2 \qquad \text{(II-2)}$$

The mean square value of the v and w velocity components are defined similarly. If the fluid flow is steady, then this quantity (which is similar to the statistical variance) is constant and is computed using time averages. If the flow is unsteady, then the turbulence intensity varies with time and must be computed from ensemble averages (see Chapter III). Turbulence intensity is normalized by dividing by the mean of the velocity at the outer edge of the boundary layer or some other suitable mean velocity. For the x, y and z velocity components, turbulence intensity are:

$$\frac{\sqrt{\overline{(u')^2}}}{U_{edge}} \quad , \quad \frac{\sqrt{\overline{(v')^2}}}{U_{edge}} \quad , \quad \frac{\sqrt{\overline{(w')^2}}}{U_{edge}} \qquad \text{(II-3)}$$

The magnitude of the normalized turbulence intensity parameter is on the order of 0.01 (1.0%) for highly turbulent flows and on the order of 0.0001 (0.01%) for laminar flows [Ref. 4]. The turbulence intensity of the free stream flow in a typical wind tunnel is 0.3%. In specially designed low turbulence wind tunnels, turbulence intensities of 0.02% are attained.

2. Turbulence Length Scale

A measure of the characteristic length scale of the turbulent flow (i.e., the average size of a turbulent eddy) is obtained by measuring the spatial correlation of the velocity between two points in the flow. This requires two or more velocity sensors at different locations in the flow. If the fluctuating velocity component, u , is small compared to the mean velocity, then the eddies do not change significantly as they pass a given point. This common situation allows the use of time correlation (autocorrelation) to characterize the turbulence length scale instead of spatial correlation (eliminating the need for multiple sensors). Autocorrelation is defined using the aerodynamicist terminology as:

$$\overline{u'(t)\ u'(t+\tau)} = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} u'_i(t)\ u'_i(t+\tau) \qquad \text{(II-4)}$$

where for steady turbulence i is a time index and for unsteady turbulence i is an ensemble index. For three dimensional measurements, autocorrelations of v and w may also be defined in a similar manner.

Though harder to understand physically, cross correlations are also important to the study of turbulent flows. These are a measure of the correlation between two different components of fluctuating velocity, u′ and v′ for example. The cross correlation is defined by:

$$\overline{u'(t)\ v'(t+\tau)} = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} u'_i(t) v_i'(t+\tau) \qquad \text{(II-5)}$$

$$\overline{v'(t)\ u'(t+\tau)} = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} v_i(t) u'_i(t+\tau)$$

where for steady turbulence i is a time index and for unsteady turbulence i is an ensemble index. The cross correlation of u and v (or any two components) for zero lag are known as the "Reynolds stress" terms, and are important to understanding the convective interaction between the turbulence and the mean flow [Ref. 4].

3. Frequency Spectra

With respect to turbulence, frequency spectra are measures of the power contained in the turbulent fluctuating velocity components at a given frequency. The various frequency spectra (auto-spectra and cross-spectra) contain the

same information contained in the various correlation measures, but transformed into the frequency domain. Frequency spectra are defined in more detail in Chapter III.

D. PREVIOUS UNSTEADY TURBULENCE INVESTIGATIONS

The boundary layer on a surface such as an airfoil can, in many situations, be sensitive to external disturbances in the flow. One example of such an external disturbance has been investigated experimentally by Howard [Ref. 2]. This investigation involves the study of the boundary layer of an airfoil immersed in a propeller slipstream (see Figure II-4). The propeller wake subjects a given point on the airfoil to periodic turbulent pulses. The propeller wake provides a convenient method to study the effect on a boundary layer of an external turbulent disturbance. The periodic nature of the propeller wake pulses allow the rapid collection of many repeated disturbances for ensemble averaging, which is necessary due to the unsteady nature of the fluid flow.

"Single-wire" hot wire anemometer velocity data were collected and boundary layer velocity profiles were constructed using a simple ensemble average of time averaged segments (see Chapter IV). A single-wire velocity sensor measures the magnitude of the velocity normal to the wire (that is $\sqrt{u^2 +}$. Howard [Ref. 2] concluded that the wing boundary layer was found to vary from the laminar to turbulent

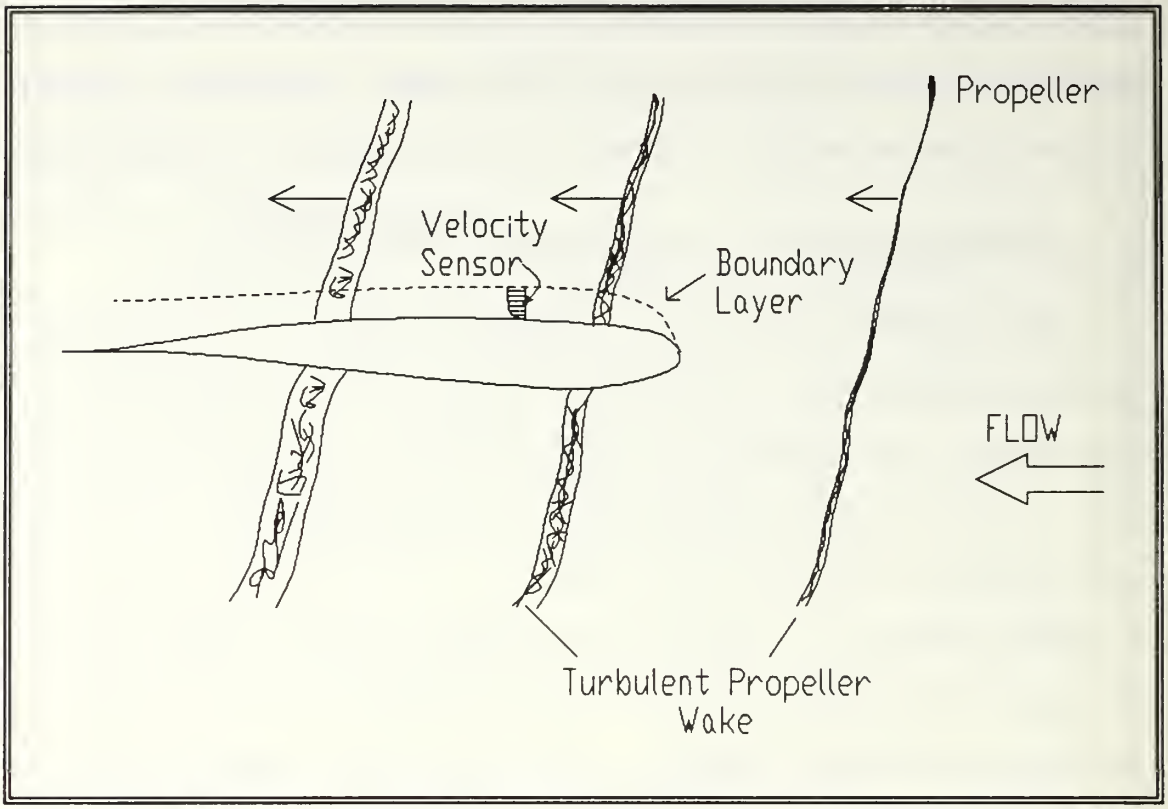

Figure II-4. AN AIRFOIL IN A PROPELLER SLIPSTREAM [Howard, Ref. 2].

condition in a periodic manner due to the passage of the propeller wake at a frequency of 20 to 40 Hz [see Figure II-5 from Howard, Ref. 2]. The length of time the turbulence remains in the boundary layer is a strong function of the pressure gradient (the change in pressure with location on the surface in the direction of mean flow). The turbulence within the boundary layer persists longer as the pressure gradient varies from favorable ($dP/dx < 0$) to adverse ($dP/dx > 0$). It was also observed that the shape of the velocity profile has a strong dependence on the instantaneous level of external turbulence. Furthermore, the instantaneous drag coefficient

of the airfoil section was found to vary from that expected for a laminar boundary layer to below that expected for a conventional fully turbulent boundary layer. In other words, the mean drag was well below that expected for a fully turbulent boundary layer. No correlation or spectral analyses were performed.

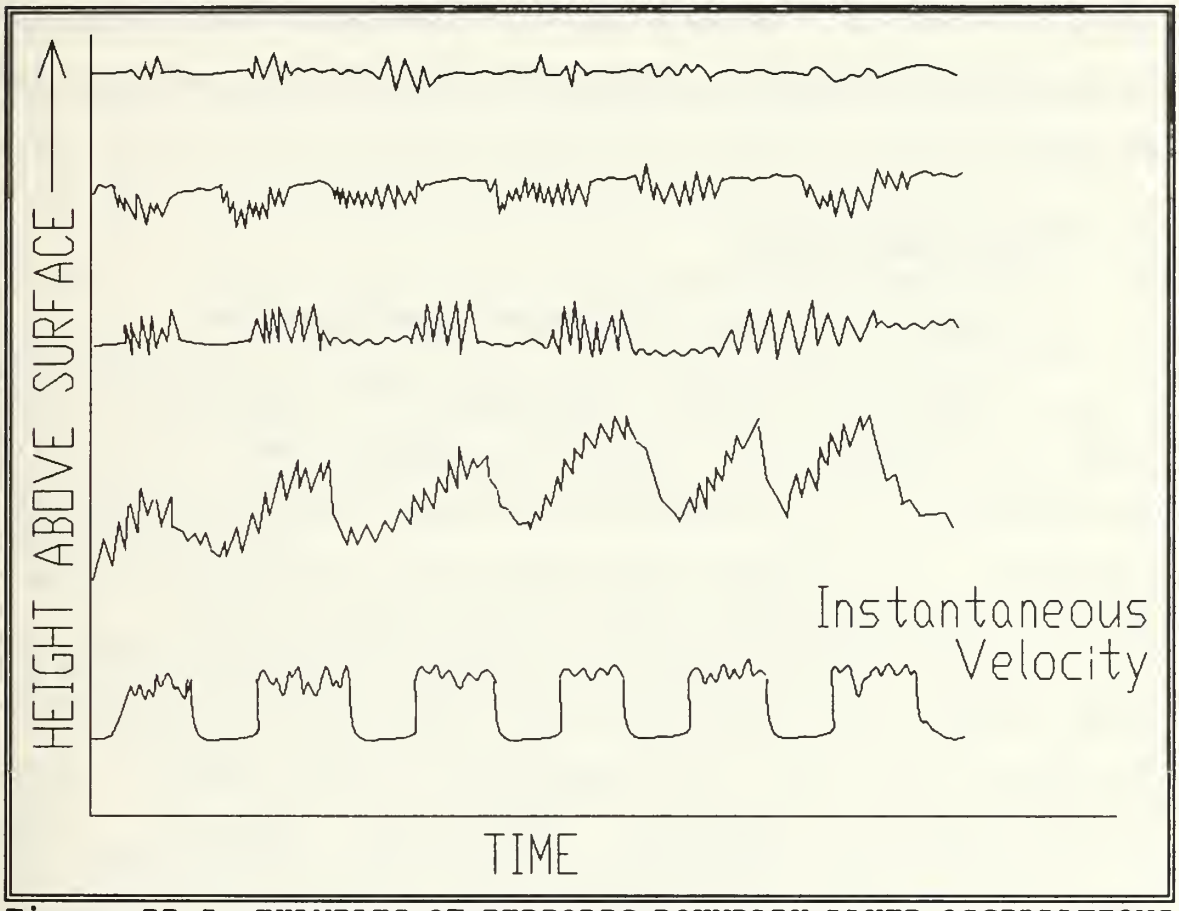

Figure II-5. EXAMPLES OF PERIODIC BOUNDARY LAYER OSCILLATIONS [Howard, Ref. 2].

An understanding of the time-dependent behavior of an airfoil boundary layer responding to external disturbances will provide basic information to studies of the use of dynamic maneuvering for increased fighter aircraft agility.

Dynamic maneuvering involves the use of large deflections of aircraft control surfaces at very high angles of attack (post-stall angles of attack) in order to achieve maximum agility particularly at low speeds. Dynamic maneuvering creates the possibility of an aircraft wing or control surface being exposed to the turbulent wake disturbance from other surfaces. Another potential application is to the extremely turbulent flow (on the order of 20%) in turbomachinery such as jet engines.

E. THESIS OBJECTIVES

1. Future Unsteady Turbulence Investigations

Future wind tunnel tests have been proposed for fall 1988 to continue the study by Howard [Ref. 2]. The investigation will be expanded to include "dual-wire" hot wire anemometer measurements which will allow resolution of the u and v velocity components within the boundary layer. Boundary layer profiles and turbulence intensity measurements will be determined for the two component velocity data. Also, autocorrelation analyses (turbulence scales), cross correlation analyses (Reynolds stresses) and spectral analyses will be accomplished using both single and dual wire velocity sensors to examine these properties in a periodically disturbed boundary layer. Of interest are the differences

between a conventional turbulent boundary layer and one excited by freestream turbulence.

2. Thesis Objectives

This thesis describes an effort to develop a set of data analysis algorithms and software (with appropriate mathematical justification) to support the unsteady turbulence investigation described in the section above. The work includes investigations into alternate methods for determining some of the statistical properties used to characterize the unsteady (non-stationary) periodically perturbed boundary layer. Where alternate algorithms are developed, the strengths and limitations of each method are examined and the "best" algorithm is recommended based on criteria such as accuracy, error sensitivity, speed of execution and ease of implementation.

Specifically, data analysis algorithms are required to extract certain statistical parameters used to characterize the nature of turbulent fluid flow from the "raw" velocity data. The specific unsteady turbulence parameters required to support the planned wind tunnel tests described above are listed as follows:

Required Turbulence Parameters

- Mean Velocity
- Fluctuating velocity component
- Turbulence intensity

- Spectral (frequency) characteristics
- Auto/cross correlation (turbulence scales)

The basic design requirements for the completed data analysis software include the following constraints:

Software Design Constraints

- Software must run on an IBM PC/AT class microcomputer.
- Software must be written in FORTRAN.
- Data processing will be performed after testing (no real time processing requirements).
- Input file formats are defined by the analog-to-digital conversion system.

III. MATHEMATICAL BACKGROUND AND DEVELOPMENT

The purpose of this chapter is to present a brief overview of the basic mathematical concepts which were applied to develop the specific data analysis methods for the unsteady turbulence investigation. The important concepts include the statistics of random processes associated with time series analysis such as time and ensemble mean and variance, autocorrelation and time cross correlation, probability distributions and stationary vs. non-stationary random processes. Also important is the Discrete Fourier Transform (DFT), particularly as it applies to spectral estimation. The following references were used extensively in the preparation of this chapter; Bendat and Piersol [Ref. 6], Brigham [Ref. 7], Strum and Kirk [Ref. 8], Hamming [Ref. 9], Marple [Ref. 10], and Otnes and Enochson [Ref. 11].

A. STATISTICS OF RANDOM PROCESSES

1. Definition of Terms

Turbulent fluid motion is characterized by random fluctuations of the observed quantities (such as velocity or pressure) with respect to space and time. Random data are not

deterministic and as such, cannot be described by explicit mathematical expressions. Statistical analysis methods are used to extract and summarize useful information that characterizes the nature of the random process. A sequence of observations or measurements is called a "sample function", a "time series" or a "time history record". Physically, turbulent motion is a continuous process. But the measurements sample the continuous process at discrete times. In addition to being discretized in time, the observed quantity (e.g. velocity) is also quantized for storage in a digital computer. For example, given a 12 bit analog to digital conversion, the measured velocity may only take on 2^{12} or 4096 distinct values within some predefined range. All of the time series considered here will be discrete and quantized and are denoted as:

$$u_i \text{ or } u(t_i), \quad i=1,2,....,N$$

A collection of sample functions is called an "ensemble" and is denoted as:

$$u_k(t_i)$$

An ensemble of all possible sample functions defines the underlying "random process". Figure III-1 illustrates an ensemble of N time history records.

2. Probability Distribution Functions

Given a discrete random variable, u_i, there exists a finite probability that u_i will be equal to a particular value

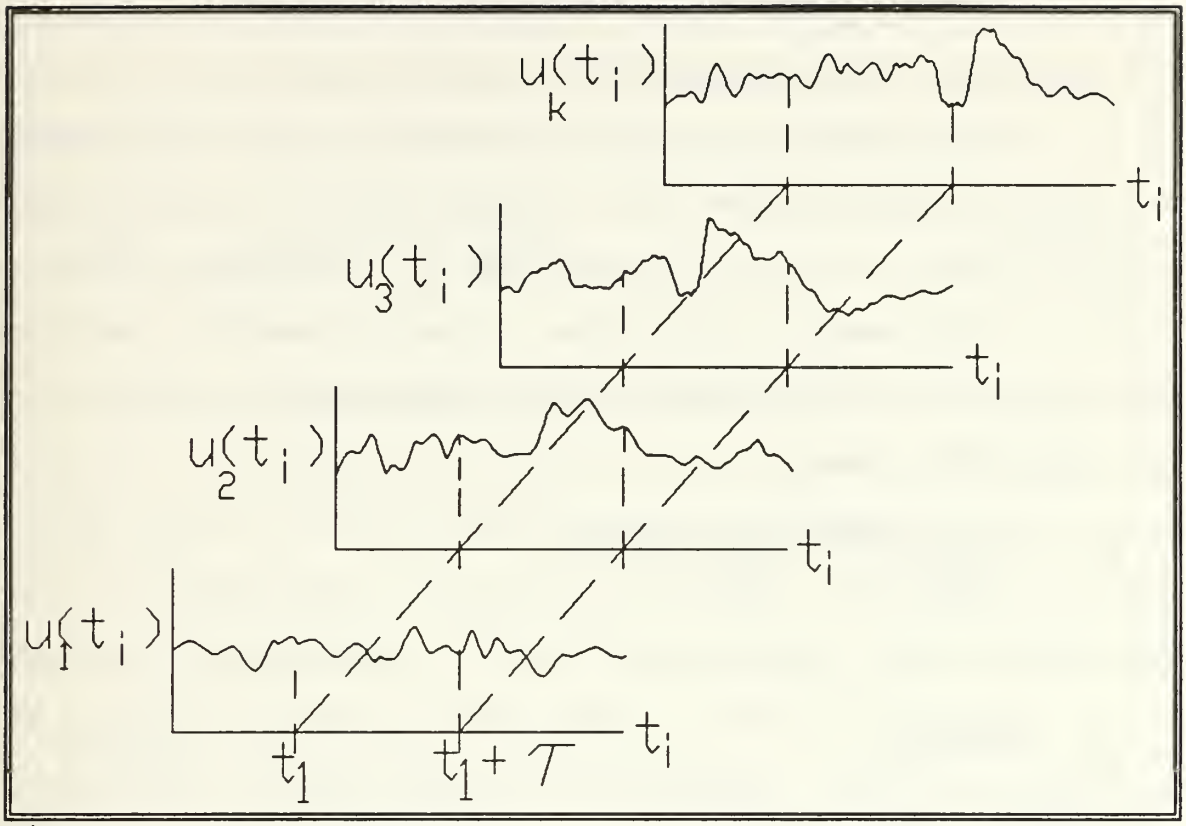

Figure III-1. AN ENSEMBLE OF TIME HISTORY RECORDS [BENDAT AND PIERSOL, REF. 6].

at time index i. For example, if u is the fluid velocity at a point in the turbulent boundary layer and the index i represents a particular instant in time (t_i), then there is a probability value (between zero and one), say 0.3 for this example, that u will be equal to a particular velocity, U, where U is one of 4096 distinct values allowed for u by its digital representation. This is expressed as:

$$p(u_i) = \text{PROB}(u_i=U) = 0.3 \qquad \text{(III-1)}$$

The probability mass function, $p(u_i)$ of a discrete random variable defines how the total probability of 1.0 is distributed over the range of possible values of the random

variable. The probability mass function is usually not known for an empirically measured set of random data.

Given a continuous random variable, u(t), there exists a finite probability that u(t) will be between two particular values U and U + Δ U at a given time t. For the continuous case, u may take on any value. The probability density function, f(u), is defined by the differential relation as Δ U approaches zero:

$$f(u) = \lim_{\Delta U \to 0} \left[\frac{\text{PROB}(U<u(t)<U+\Delta U)}{\Delta U} \right] \qquad \text{(III-2)}$$

Note that,

$$\int_{-\infty}^{\infty} f(u)\,du=1 \qquad \text{(III-3)}$$

For many practical applications, a continuous random process which has been quantized for a digital computer (such as the velocity example above) will still be discussed in terms of its continuous probability density function even though it actually takes on only discrete values. This is a valid approximation as long as the quantization is sufficiently "fine". (At least 512 to 1024 distinct values over the range of the measured quantity will be considered sufficient.)

There are many theoretical probability distribution functions or "distributions" which have been defined for both discrete and continuous random variables. The most common is the continuous "Gaussian" or "Normal" distribution. If a

random variable u has mean μ_u and variance σ_u^2 (see definitions below), then it is normally distributed if its probability density function is given by:

$$p(u) = \frac{1}{\sqrt{2\pi\sigma_u^2}} \text{EXP}\left[-\frac{(u-\mu_u)^2}{2\sigma_u^2}\right] \qquad \text{(III-4)}$$

The normal distribution appears often in nature and much of its importance is due to the Central Limit Theorem, which states that the distribution of a sufficiently large linear combination of finite variance independent random variables approaches a normal distribution. This holds even if the distribution of each member of the independent linear combination is not normally distributed. This is important because an empirically measured random process can often be modeled as a linear combination of many independent random variables (say 20 or 30 as a rule of thumb). Then by the Central Limit Theorem, the distribution of the measured random process may be assumed normal. If a random variable is normal, then the mean and variance provide a complete description of the nature of the random process. Bendat and Piersol [Ref. 6] presents a good summary of many other continuous and discrete probability density functions and their uses, as well as a more detailed discussion of the Central Limit Theorem.

3. Mean Value

The most basic statistic is the mean value. The mean value is also known as the expected value or average value. In terms of the discrete random variable u and the probability mass function $p_k(u)$, the mean value of the ensemble of time history records, $u_k(t_i)$ at a fixed time t_i is defined by the equation:

$$\mu_u(ti) = E[u_k(t_i)] = \sum_{k=1}^{\infty} u_k(t_i)p(u) \qquad \text{(III-5)}$$

where the "E[ ]" notation is the expected value of u. The expected value operator may also be applied to any function of a random variable and is defined by:

$$E[g(u_k)] = \sum_{k=1}^{\infty} g(u_k)p(u) \qquad \text{(III-6)}$$

In terms of the probability density function f(u), and the continuous random variable u(t), the mean (expected) value of the ensemble of time history records at fixed time t is defined by:

$$E[u_k(t)] = \int_{-\infty}^{\infty} u_k(t)\ f(u)\ du \qquad \text{(III-7)}$$

For practical measurements, the number of sample functions (and the length of time) are finite. The sample mean (an unbiased estimate of the true mean) of a finite ensemble of N time history records at a fixed time t_i is defined by:

$$\hat{\mu}_u(t_i) = \bar{u}(t_i) = \frac{1}{N} \sum_{k=1}^{N} u_k(t_i) \qquad \text{(III-8)}$$

where the ^ notation denotes an estimator of the true parameter. If a random process is a member of the special class of "ergodic" random processes (see below), then the ensemble averaged mean value of the random process is invariant with time and is equal to the time averaged mean value of a single (infinite) time history. The time mean of a single time series is defined by:

$$\mu_u(k) = \lim_{M\to\infty} \frac{1}{M} \sum_{l=1}^{M} u_k(t_l) = \mu_u(t_l) \qquad \text{(III-9)}$$

For a practical finite time history record, the time sample mean (again, an unbiased estimate of the true mean for ergodic random data) is given by the equation:

$$\hat{\mu}_u(k) = \bar{u}(k) = \frac{1}{M} \sum_{l=1}^{M} u_k(t_l) \qquad \text{(III-10)}$$

4. Variance

Another important statistic is the "variance" of the random process. The variance is a measure of the mean square deviation or dispersion of the data about its mean. The square root of the variance is called the standard deviation. The variance of an ensemble of discrete time history records is defined by the equation:

$$\sigma_u^2(t_l) = E\left[\left(u_k(t_l)-\mu_u(t_l)\right)^2\right] \qquad \text{(III-11)}$$

$$= \sum_{k=1}^{\infty}\left[\left(u_k(t_l)-\mu_u(t_l)\right)^2 p(u)\right]$$

where the mean value, μ_u, is computed by equation III-5 and σ_u is called the standard deviation. For a practical finite ensemble, an unbiased estimate of the true variance is given by the sample variance:

$$\hat{\sigma}_u^2(t_1) = S_u^2(t_1) = \frac{1}{N-1} \sum_{k=1}^{N} \left[u_k(t_1) - \bar{u}(t_1) \right]^2 \qquad \text{(III-12)}$$

Similar to the discussion of mean above, if a random process is ergodic (see below) then the ensemble variance is equal to the variance of a single infinite time history record and is constant with time. The time variance is defined by:

$$\sigma_u^2(k) = \lim_{M \to \infty} \frac{1}{M-1} \sum_{i=1}^{M} \left[u_k(t_i) - \mu_u(k) \right]^2 = \sigma_u^2(t_1) \qquad \text{(III-13)}$$

where the mean value is computed by equation III-9. For a practical finite time history record, an unbiased estimate of the time variance is given by the time sample variance:

$$\hat{\sigma}_u^2(k) = S^2(k) = \frac{1}{M-1} \sum_{i=1}^{M} (u_k(t_i) - \bar{u}_k)^2 \qquad \text{(III-14)}$$

5. Autocorrelation and Cross Correlation Functions

Correlation is a measure of the degree of linear relationship that exists between two random variables, u and v, at a given instant in time. The correlation function introduces another variable, τ, which is a time lag between the two random variables. The correlation function essentially gives a correlation measure for each of the possible values of the lag τ (that is, the amount of

correlation when one time series leads or lags the other). When u equals v, the correlation function is called autocorrelation. When u is different than v, the correlation function is called cross correlation.

In terms of the expected value operator, the autocorrelation of a real valued time series is defined as:

$$R_{uu}(t_i, t_i+\tau) = E\left[u_k(t_i)u_k(t_i+\tau)\right] \qquad \text{(III-15)}$$

For an ergodic random process, the ensemble definition of autocorrelation is equal to the time averaged definition for a single infinite duration time series. The time averaged definition of autocorrelation is:

$$R_{uu}(\tau) = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{i=-N}^{N} u(t_i)u(t_i+\tau) \qquad \text{(III-16)}$$

An unbiased estimate of the autocorrelation function of a real, discrete, finite, ergodic random process with zero mean $u(t_i)$ is given by:

$$\hat{R}_{uu}(\tau) = \frac{1}{N-r} \sum_{i=1}^{N-r} u(t_i)u(t_i+\tau) \qquad r=0,1,2,\ldots,m \qquad \text{(III-17)}$$

where r, the lag number, is defined by τ such that $\tau = r\Delta t$, Δt is the sample interval and m is the maximum lag number. The maximum lag number, m, can go up to N-1, but for typical computations is usually much less than N (on the order of 1/10 N). This equation is expressed only for positive lag numbers. However, it has been shown that the autocorrelation function

is an even function; thus $R_{uu}(\tau)$ equals $R_{uu}(-\tau)$. See Bendat and Piersol [Ref. 6].

For two random processes u and v, the cross correlation function is defined by:

$$R_{uv}(t_i, t_i+\tau) = E\left[u_k(t_i)v_k(t_i+\tau)\right] \qquad \text{(III-18)}$$

$$R_{vu}(t_i, t_i+\tau) = E\left[v_k(t_i)u_k(t_i+\tau)\right] \qquad \text{(III-19)}$$

An unbiased estimate of the cross correlation function for a practical finite data set consisting of two ergodic random variables with zero mean is given by:

$$\hat{R}_{uv}(\tau) = \frac{1}{N-r}\sum_{i=1}^{N-r} u(t_i)v(t_i+\tau) \qquad \text{(III-20)}$$

$$\tau = r\Delta t$$

$$r = 0, 1, 2, \ldots, m$$

$$\hat{R}_{vu}(\tau) = \frac{1}{N-r}\sum_{i=1}^{N-r} v(t_i)u(t_i+\tau) \qquad \text{(III-21)}$$

$$\tau = r\Delta t$$

$$r = 0, 1, 2, \ldots m$$

The cross correlation function estimate is in general neither odd or even, but for negative lag numbers, it can be shown to satisfy the relation that, for real valued data, $R_{uv}(\tau)$ equals $R_{vu}(-\tau)$. The autocorrelation function is a special case of the cross correlation function for u equal v. The auto correlation and cross correlation functions may be estimated directly using equations III-17 and III-20, III-21 or may be computed indirectly using Fourier transform methods which are computationally more efficient for large data sets. Fourier transform methods are addressed in section III-B.

6. Spectral Density Functions

The autospectral density function is a measure of the energy content of a time history record at a given frequency. The classical definition of the autospectral density function of a discrete random variable u, is the Fourier transform of the autocorrelation function. In terms of the discrete Fourier transform, the autospectral density function is defined by:

$$S_{uu}(f_k)=\Delta t \sum_{i=-\infty}^{\infty} R_{uu}(i\Delta t)\,EXP[-j2\pi ik/N] \qquad \text{(III-22)}$$

$$k=0,\pm 1,\pm 2,\ldots$$

where f_k is a particular frequency, t is the sample interval and I is the sample index such that = i t. By the Wiener-Khinchine relations [Ref. 6] it has been proved that the corresponding spectral density functions and correlation functions constitute Fourier transform pairs. For a real valued time series, equation III-22 may be simplified to yield the "single sided" spectral density function:

$$G_{uu}(f_k)=4\Delta t \sum_{i=-\infty}^{\infty} R_{uu}(i\Delta t)\cos(2\pi ik/N) \qquad \text{(III-23)}$$

$$k=0,1,2,\ldots$$

which is only defined for positive frequencies. Equation III-23 is therefore twice the magnitude of the even function "two sided" spectral density function given by equation III-22. For practical finite length time series, substitute the autocorrelation function estimate (equation III-16) into equation III-23. The importance of this topic warrants a more

detailed discussion. Additional material will be included in the review of the Fourier transform methods in Section III-B.

7. Classes of Random Data

Random processes may be classified as either stationary or non-stationary (see Figure III-2). Stationary data are further classified as ergodic or non-ergodic. Non-stationary data are classified as non-stationary mean, non-stationary variance, non-stationary frequency and other specialized classes. A random process is non-stationary (the most general case) when the ensemble averaged mean and autocorrelation function (and other ensemble properties) vary with time. The special classes of non-stationary data, non-stationary mean, variance and frequency, imply that the mean, variance or frequency content of the ensemble vary with time, but the other ensemble averaged properties are time invariant. In the most general case of non-stationary data, "everything" varies with time.

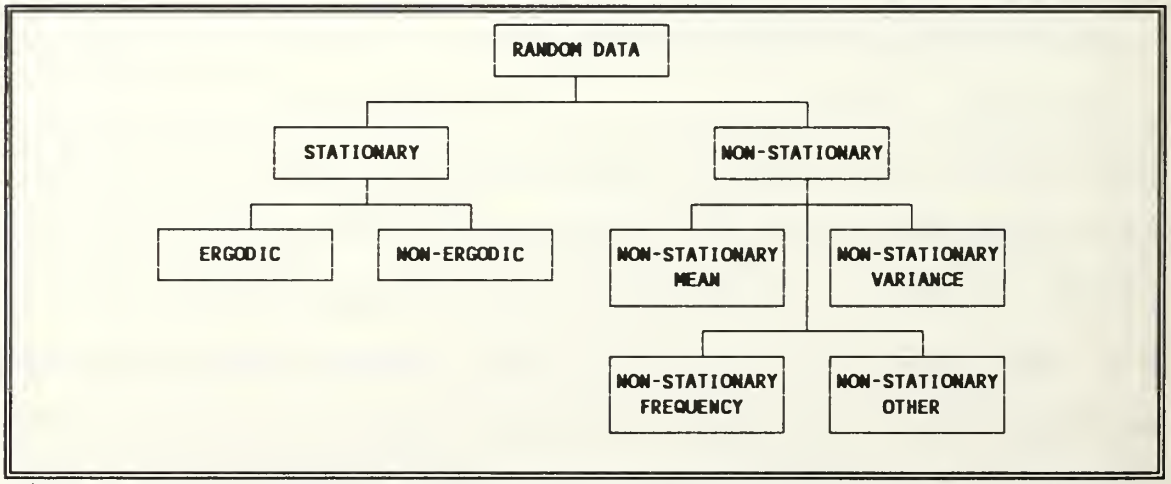

Figure III-2. RANDOM DATA CLASSIFICATIONS [BENDAT AND PIERSOL, REF. 6].

A random process is classified as "weakly stationary" when the ensemble mean and the autocorrelation function are time invariant. "Strongly stationary" refers to the special class of data where all ensemble averaged statistical properties (moments) are time invariant.

A random process is classified as "ergodic" if it is stationary and if the time averaged mean value and autocorrelation function (and all other time averaged properties) are equal to the corresponding ensemble averaged values. Thus for an ergodic random process, it is possible to derive desired statistical information about the entire random process from a single time history record of sufficient duration. The mathematics associated with the analysis of random processes is much simpler if the process is ergodic.

B. DISCRETE FOURIER TRANSFORM METHODS

1. Definition of Terms

A physical process (in general, either a random or deterministic process represented by real or complex numbers) is typically described in the "time domain". For example, the turbulent velocity measurements u(t) where the velocity is described as a function of time (i.e., the typical time history record). A physical process may also be described in the "frequency domain" where the velocity is given by its amplitude, U(f), as a function of its frequency, f. In

general, the amplitude is also a complex number. U(f) and u(t) are essentially two different representations of the same data record. Both representations contain exactly the same information regarding the process, presented differently. A method for transforming between these two representations is the "fourier transform" and the "inverse fourier transform".

Fourier transforms are defined for both continuous and discrete processes. The planned wind tunnel tests described in Chapter II involve only discretized data. Therefore, the discussion here is limited to discrete (sampled) data. Moreover, most modern time series analyses are accomplished on discretized data to take advantage of the power and speed of digital computers. Some specific issues associated with sampled data will be discussed before reviewing the discrete fourier transform.

2. Data Sampling

For a continuous process u(t) which is discretized or sampled at equal time intervals, the sampling interval, Δt, is the amount of time between samples. The sampling rate, f_s, is the inverse of the sampling interval and is usually expressed in units of Hertz (Hz) or samples per second (SPS):

$$f_s = \frac{1}{\Delta t} \qquad \text{(III-24)}$$

The "Sampling Theorem" (see Strum and Kirk [Ref. 8] and Brigham [Ref. 7]) states that if a continuous signal,

u(t), contains no frequency components above a frequency f_{max}, then the signal can be uniquely represented by equally spaced samples $u(t_i)$ only if the sampling frequency f_s is greater than 2 f_{max}. Furthermore, the original signal u(t) can be recovered exactly from the sampled signal by the formula:

$$u(t) = \Delta t \sum_{i=-\infty}^{\infty} u_i \frac{\sin\left[2\pi f_c(t_i - i\Delta t)\right]}{\pi(t_i - i\Delta t)} \qquad \text{(III-25)}$$

where f_c is the Nyquist critical frequency defined by:

$$f_c = \frac{1}{2\Delta t} = \frac{f_s}{2} \qquad \text{(III-26)}$$

Unfortunately, many realizable processes may contain a very wide band of frequency components. This may cause the required minimum sample rate of $2f_{max}$ to be impractically high. The phenomenon of "aliasing" is a result of sampling a wideband process. Data aliasing causes frequency information at frequencies above the Nyquist frequency to be "folded" spuriously down into the frequency range between $-f_c$ and f_c. The Nyquist critical frequency, f_c, is the folding frequency. In other words, the sample rate, f_s, must be greater than twice the frequency of the highest frequency present in the data to preclude aliasing. An alternate method to prevent aliasing is to low pass filter the continuous (analog) signal before sampling to remove all high frequency components of the data above the frequency range of interest. Then, the data

are sampled at a frequency of greater than twice the maximum frequency of interest.

3. The Discrete Fourier Transform

To transform an N point discrete time series u_i from the time domain to the frequency domain U_k (in general, both are complex), the discrete Fourier transform (DFT) is defined by:

$$U_k = \sum_{i=1}^{N} u_i \, EXP(-j2\pi ik/N) \qquad \text{(III-27)}$$

Here k is the frequency index running from -N/2 to N/2 (or 0 to N-1) and N is the total number of points in the time series u_i. The DFT is said to map u_i into U_k. If u_i is considered to be a sequence of samples of a continuous function u(t), then the DFT times the sampling interval is an approximation of the continuous Fourier transformation of u(t). That is,

$$U(f_k) \cong \Delta t \, U_k = \Delta t \sum_{i=1}^{N} u_i \, EXP(-j2\pi ik/N) \qquad \text{(III-28)}$$

where $f_k=k/N$ t and f_k varies between $-f_c$ and f_c. $U(f_k)$ is an approximation of the continuous Fourier transform because the sampled time series is of finite duration and finite frequency bandwidth (a sampled and truncated approximation of the infinite duration continuous signal). In the development of the DFT as a special case of the continuous Fourier transform, it is assumed that the finite time series (u_i) is a single period of an infinite real periodic function. This makes the

"infinite periodic function" an even function so that by the properties of symmetry, the DFT is also an even function. That is, U(f) equals U(-f). Also of importance is another symmetry property which states that if u_i is real and even (and if it is a finite length sampled version of a continuous function, it must be even), then the DFT will also be real and even (see Press, et al, [Ref. 12] and Brigham, [Ref. 7]).

The inverse discrete Fourier transform (IDFT) is defined by the following equation:

$$u_i = \frac{1}{N} \sum_{k=1}^{N} U_k \ EXP(j2\pi ik/N) \qquad \text{(III-29)}$$

The IDFT transforms a frequency domain function back into the time domain. Given U_k, by equation III-29, the IDFT will recover exactly the u_i's that were input into the DFT equation. Note that the summation is over the frequency index k for the IDFT instead of the time index i. Equations III-27 and III-29 form a Fourier transform pair. Given $U(f_k)$ (equation III-28) the IDFT is defined by:

$$u_i = \frac{1}{N\Delta t} \sum_{k=1}^{N} U(f_k) \ EXP(j2\pi ik/N) \qquad \text{(III-30)}$$

The only difference is the factor of $1/\Delta t$ which causes the Fourier transform pair III-20 and III-25 to depend on the time scale while the pair III-19 and III-21 do not. The key point is to be consistent in the Fourier transform pair used.

4. Correlation Functions

Correlation functions (auto and cross) were defined in section III.A.5. above. In this section, the "indirect" DFT method for computing the auto or cross correlation functions is described. The Discrete Correlation Theorem Ref. 7] states that the "circular" correlation functions may be determined in the frequency domain through the following relationship:

$$\hat{R}^{c}_{uv}(\tau) = \sum_{i=1}^{N} u(t_i)v(t_i+\tau) = \frac{1}{N}\sum_{k=1}^{N} U^{*}_{k}V_{k}EXP(j2\pi rk/N) \qquad \text{(III-31)}$$

$\tau = r\Delta t$

$r = 0,1,2,\ldots,N-1$

In words, the circular correlation function of two discrete time series $u(t_i)$ and $v(t_i)$ may be computed by multiplying the complex conjugate of the DFT of the first time series by the DFT of the second time series, and then computing the IDFT of the product. This is essentially a linear convolution. Stated another way, the expression $U^{*}_{k}V_{k}$ forms a Fourier transform pair with the left hand side of equation III-31. (Again, when u equals v then equation III-31 yields the autocorrelation.)

The nature of the circular correlation function results in the correlation values at positive and negative lags being summed. To obtain the standard correlation functions (equations III-17, III-20, and III-21) instead of the circular correlation, add N zeros to the end of u and v

(or at least m zeros, where m equals the maximum lag number r). Then compute the circular correlation in the normal way. The added zeros will cause the negative lags to be separated from the positive lags [Ref. 6]. This comes at a cost since the two time series are now twice as long. The standard correlation function estimate is given by:

$$\hat{R}_{uv}(\tau) = \frac{N}{N-r}\hat{R}^{c}_{uv}(\tau) \quad \tau=r\Delta t, \quad r=0,1,2,\ldots,N-1 \qquad \text{(III-32)}$$

This method, while more complicated, is more computationally efficient for large data sets because it can take advantage of an efficient Fast Fourier transform (FFT) algorithm [Ref. 6].

5. Autospectral Density Functions

A theorem important to the development of Fourier transform spectral analysis techniques is Parseval's Theorem. Parseval's Theorem states that the energy contained in the time domain of a time series u_i is equal to the energy contained in the signal's frequency domain [Ref. 7]. In terms of the discrete transform, Parseval's Theorem is stated as follows:

$$\sum_{i=1}^{N} |u_i|^2 = \frac{1}{N}\sum_{k=1}^{N} U_k^* U_k \qquad \text{(III-33)}$$

This makes intuitive sense because the information contained in the time domain representation of a signal is equivalent to that contained in the frequency domain.

As described in section III.A.6. above, the classical definition of the spectral density function is the Fourier transform of the autocorrelation function. An alternative method applies the Fourier transform directly to the time series to obtain a spectral estimate. When an FFT algorithm is used to compute the Fourier transform, the spectral estimate is obtained many times faster than the classical method. (The FFT algorithm will be discussed in the next section.) For an infinite length time series, the spectral density function is defined by:

$$S_{uu}(f_k)=\lim_{N\to\infty} E\left[\frac{1}{(2N+1)\Delta t}\left|\Delta t\sum_{i=-N}^{N} u_i EXP(-j2\pi ik/N)\right|^2\right] \qquad \text{(III-34)}$$

or

$$S_{uu}(f_k)=\lim_{N\to\infty} E\left[\frac{1}{(2N+1)\Delta t}U_k^*(f_k)U_k(f_k)\right]$$

and $U(f_k)$ is defined by equation III-28. Bendat and Piersol [Ref. 6] have shown that equation III-34 is equivalent to equation III-22. For a practical finite length time series, the basic "raw" sample spectral density function may be estimated using the equation:

$$S_{uu}(f_k) = \frac{1}{N\Delta t}U^*(f_k)U(f_k) \qquad \text{(III-35)}$$

This is the original periodogram power spectral density (PSD) estimator. Marple [Ref. 10] shows that this PSD estimator will yield unstable PSD estimates which possess a variance of

the same order as the power estimate. Several averaging techniques have been developed to improve the spectral estimator. The specific technique employed for this thesis will be discussed in the next chapter where details of the specific algorithm are presented.

6. Windows in Spectral Estimation

Another concept of importance to spectral estimation is that of window functions. A typical window function is a function which starts at zero and smoothly rises to a peak value of one, then smoothly falls back to zero. The need for window functions or "windowing" arises from estimating autospectral density functions using finite length time series. When a Fourier series is used to represent a function, an exact representation of the function can be obtained, in general, only when an infinite number of terms in the Fourier series are retained. When the Fourier series is truncated, the function is approximated. Similarly, the discrete autocorrelation function values can be thought of as the coefficients of a Fourier series representing the spectral density function. When a time series (and its autocorrelation function) is of finite length (truncation), then the resulting spectral density function is an approximation of the true spectral density function due to the truncated Fourier series. Window functions are used to improve the approximation of the spectral density function.

Recall that the spectral density function (equation III-34) is defined for time samples running in the limit from minus infinity to plus infinity. When a practical finite length time series is used (equation III-35), this is analogous to the coefficients of an infinite Fourier series being multiplied by a function which is zero valued outside of the length of the finite sample record (i.e., is zero before $t = 0$ and after $t = N\Delta t$) and is one during the length of the time history record. Such a window function is known as the "rectangular window". This usually introduces a discontinuity at the beginning and end of the time series as the DFT "sees" it. Discontinuous functions are especially difficult for a truncated Fourier series to approximate. The Fourier coefficients corresponding to the higher frequencies are needed for the Fourier series to properly represent the discontinuous function. But the high frequency coefficients are discarded when the Fourier series is truncated.

Windowing is designed to improve the spectral density function approximation. This is accomplished by multiplying the input time series by a window function which starts at zero, smoothly rises to a peak of one, then smoothly falls back to zero. This "smoothes out" the discontinuities at the beginning and end of the time series record. Then the high frequency coefficients of the Fourier series are smaller in magnitude and have a much less significant effect on the

Fourier series representation of the spectral density function. Thus when the Fourier series is truncated, less information is discarded and a better approximation is the result.

Figure III-3 presents an example of the resulting spectral estimate of a 3750 Hz sine wave (amplitude of one) sampled at 15000 Hz. Thus the 3750 Hz signal is centered on a frequency bin (i.e., the time series contains an integer number of sine wave periods). The resulting spectral estimate is exact because the spectral density function is represented by a Fourier series with a single non-zero coefficient. Figure III-4 presents the more general case where the time history record is not an integer number of sine wave cycles, in this case a sine wave of 3700 Hz. The 3700 Hz signal falls between two frequency bins. Note that the power contained in the sine wave has now "leaked" into nearby frequency bins. In fact, all of the frequency bins contain some of the power that should appear only at 3700 Hz.

Figure III-5 presents the spectral estimate of the 3700 Hz sampled sine wave time series which has been previously multiplied by a "von Hann" window (equation III-36). Note that leakage has been reduced to a few adjacent frequency bins yielding a better approximation to the exact spectral density function. The magnitude of the power at each frequency has been corrected to account for the effect of the

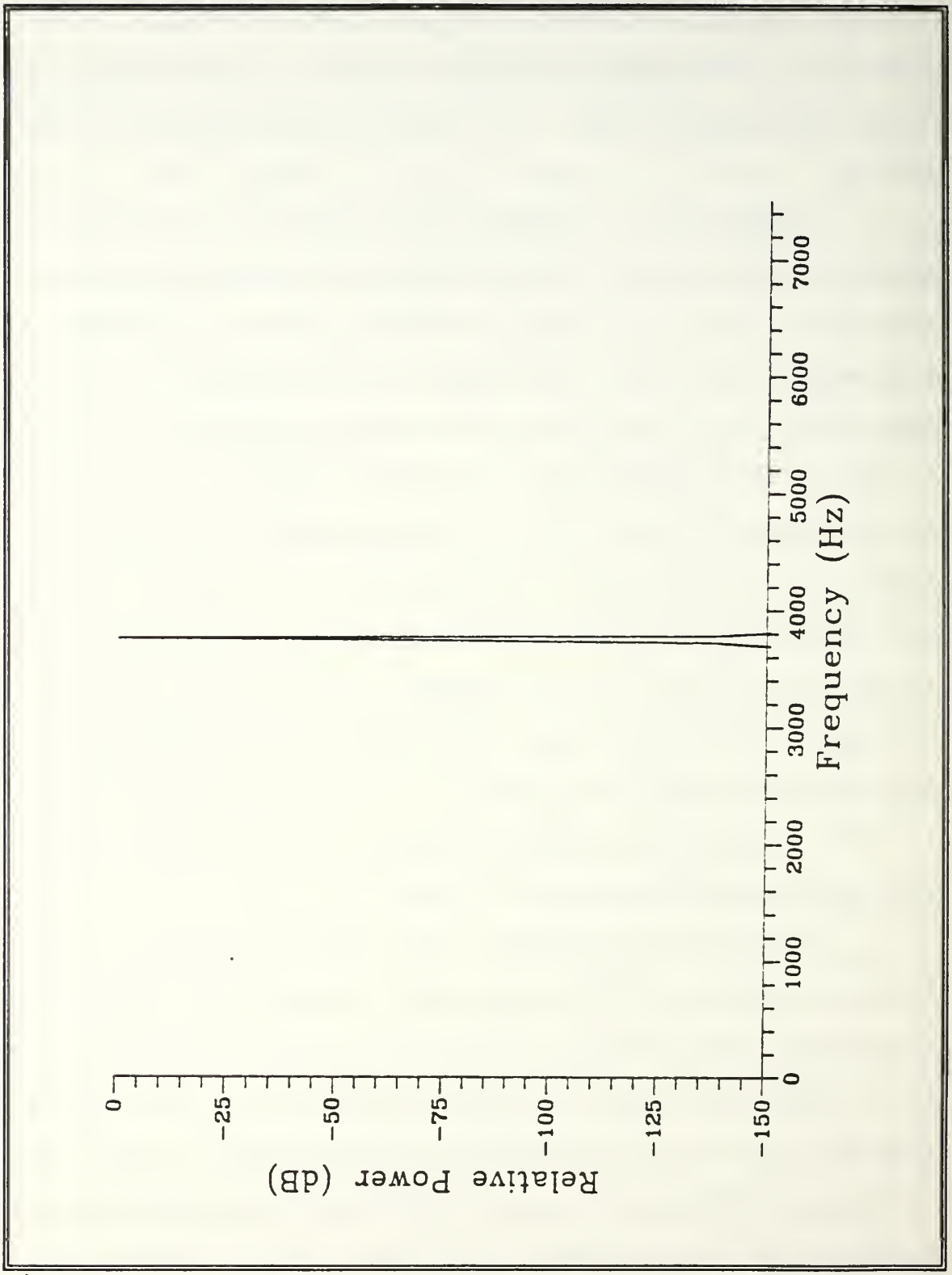

Figure III-3. SPECTRAL ESTIMATE OF 3750 HZ SINE WAVE SAMPLED AT 15,000 HZ, NO WINDOW FUNCTION.

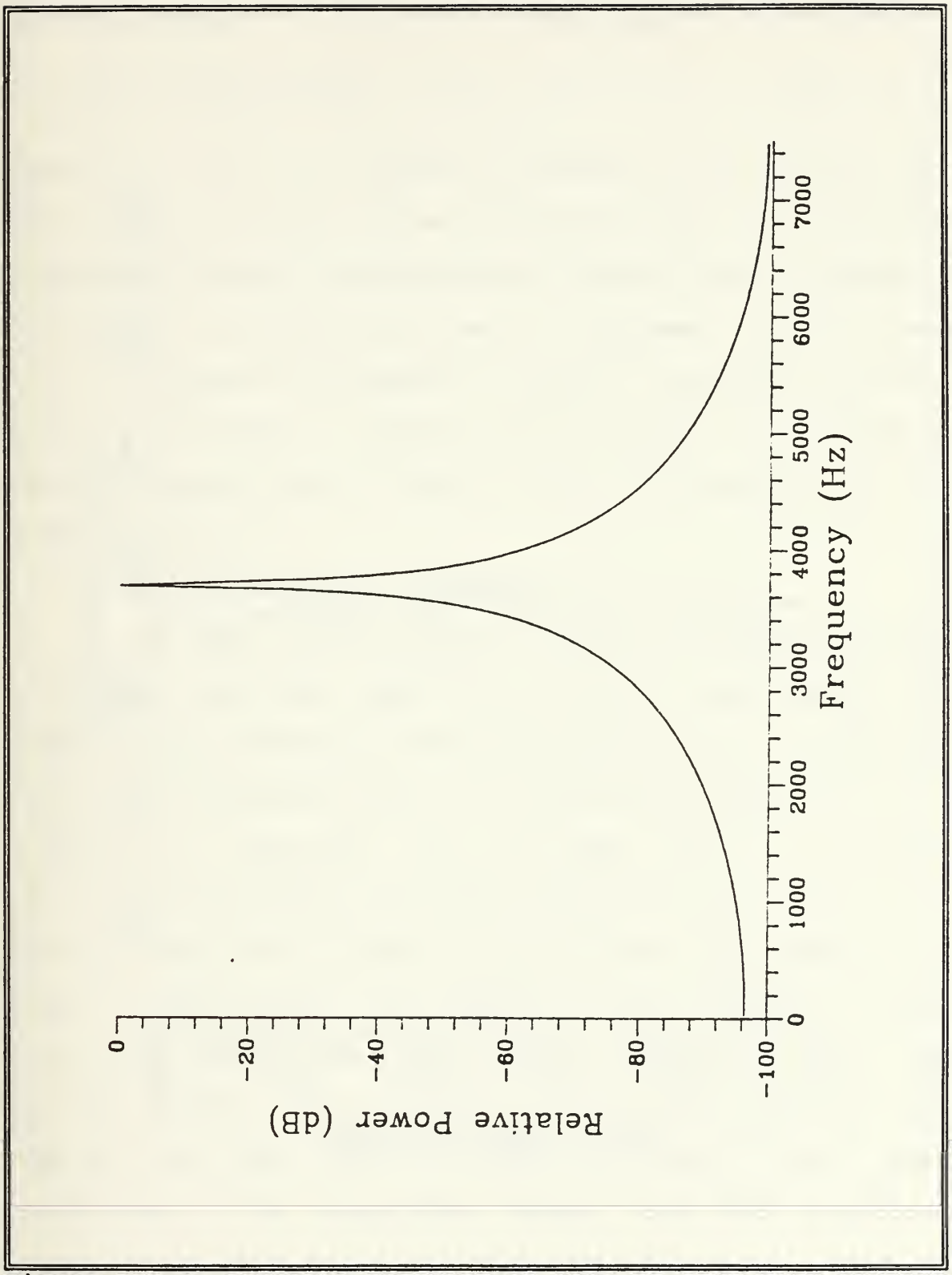

Figure III-4. SPECTRAL ESTIMATE OF 3700 HZ SINE WAVE SAMPLED AT 15,000 HZ, NO WINDOW FUNCTION.

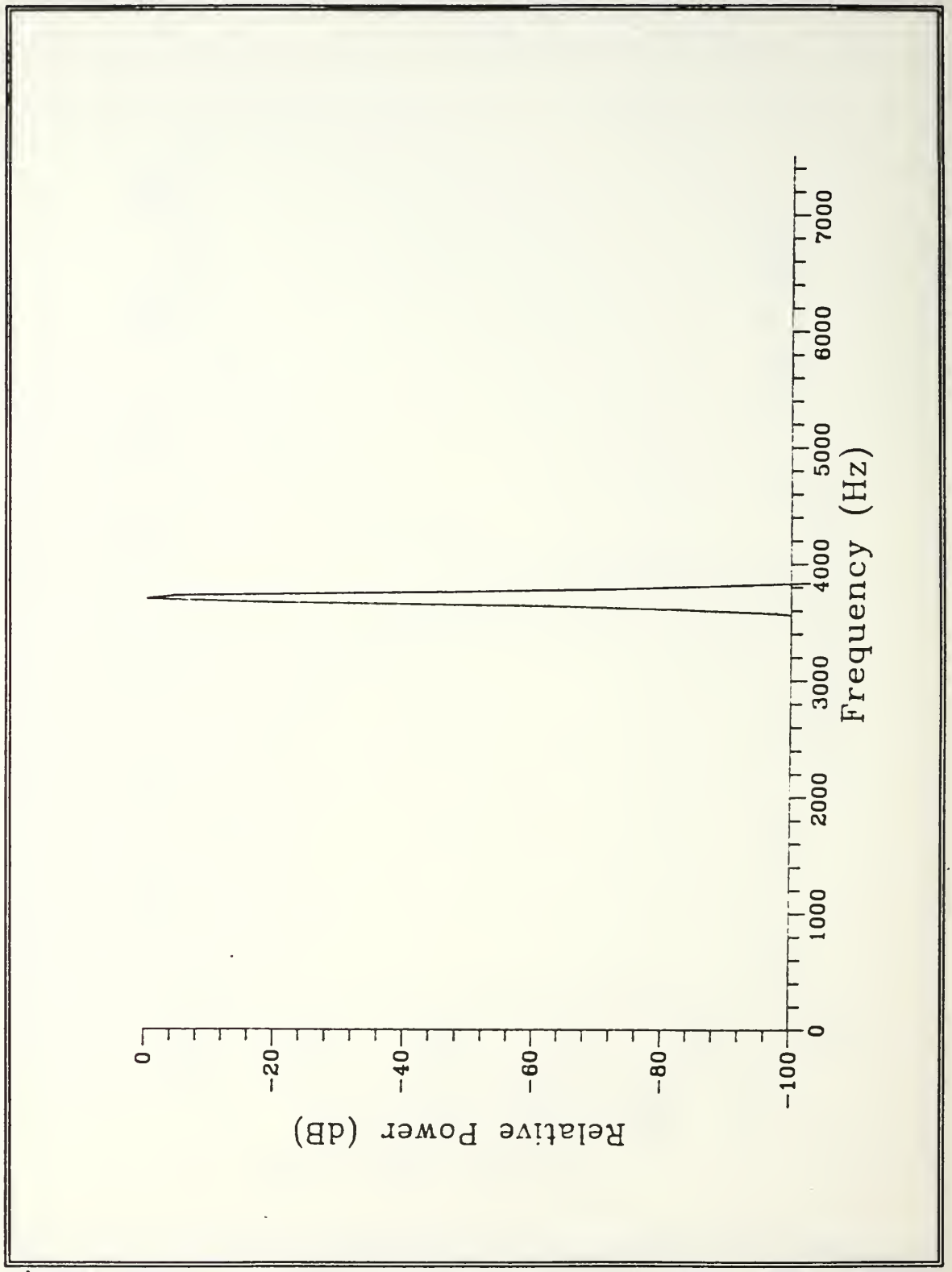

Figure III-5. SPECTRAL ESTIMATE OF 3700 HZ SINE WAVE SAMPLED AT 15,000 HZ, VON HANN WINDOW FUNCTION.

window function. The von Hann window function is given by:

$$w(t_i)=0.5-0.5\cos\left(\frac{2i}{N}\pi\right) \quad i=1,2,\ldots,N \qquad \text{(III-36)}$$

where N is the total number of points in the time series. There are many other window functions which have different frequency response characteristics and are used to optimize the spectral estimator for specific purposes. Some of these will be discussed in Chapter IV with the discussion of the specific algorithms. A paper by Harris [Ref. 13] provides an excellent description of the use and characteristics of many window functions.

7. The Fast Fourier Transform

The fast Fourier transform (FFT) is a computationally efficient algorithm (actually several algorithms) for computing the discrete Fourier transform given by equation III-27. If equation III-27 is implemented explicitly on a computer, it requires on the order of N^2 operations (multiplies and adds) to compute the DFT of an N point time series. The FFT algorithm arrives at the same result in N $\log_2 N$ / 2 operations. For example, a time series of length 1024 would require over one million operations using the explicit method. But only about five thousand operations are required for the FFT, a factor of about 1/200 fewer operations! In simplified terms, the FFT algorithm accomplishes this by considering the DFT as a system of

equations written in matrix form. When the proper matrix factorization is applied, zeros are introduced into the factors which allow a reduction in the number of required computations. The larger the matrix system, the greater the reduction in required operations. See Brigham [Ref. 7] for a detailed description of the basic FFT algorithm.

IV. DATA ANALYSIS ALGORITHMS FOR UNSTEADY TURBULENCE

Ensemble averaging techniques may be used to determine the properties of non-stationary data when many repeated time history records are available. A general method does not exist for analyzing the properties of all types of non-stationary data from single time history records [Ref. 6]. For single time history records, special techniques must be developed that apply to limited classes of non-stationary data.

The problem addressed by this thesis falls somewhere between the two extremes of a single data record vs. many time history records. The periodic nature of the data allows ensemble averaging techniques to be employed. However, practical considerations such as computer memory and data storage limitations preclude collecting "many" time history records. That is, for the current applications, collecting 50 time history records is feasible but collecting 500 time history records is not. However, 50 time history records may not be sufficient to obtain "good" estimates of the statistical properties using strictly ensemble averaging techniques.

This chapter describes the algorithms which were developed to estimate the statistical properties of unsteady (non-stationary) turbulent fluid flow. Both ensemble averaging and special (non-stationary empirical model) techniques have been employed, often in combination. Each of the algorithms has been implemented in the FORTRAN language, and examples are presented using a representative test case data set.

A. TEST CASE DATA--UNSTEADY TURBULENT FLUID FLOW

Figures IV-1 through IV-3 present three sample records (instantaneous velocity vs. time) from an ensemble of 41 sample records recorded at a single location above the surface in the boundary layer of an airfoil. Recall that a single sample record represents one propeller revolution (called a "pulse") and two propeller blade wake passages. This typical ensemble of 41 propeller pulses will be used as the test case for most of the unsteady turbulence algorithms presented in this thesis. The data were obtained from Howard [Ref. 2] and are representative of the type of data which will be collected during future tests. These data were measured using a single-wire velocity sensor so that the velocity measurements represent the magnitude of the velocity vector in a plane normal to the wire:

$$\sqrt{u^2(t)+v^2(t)}$$

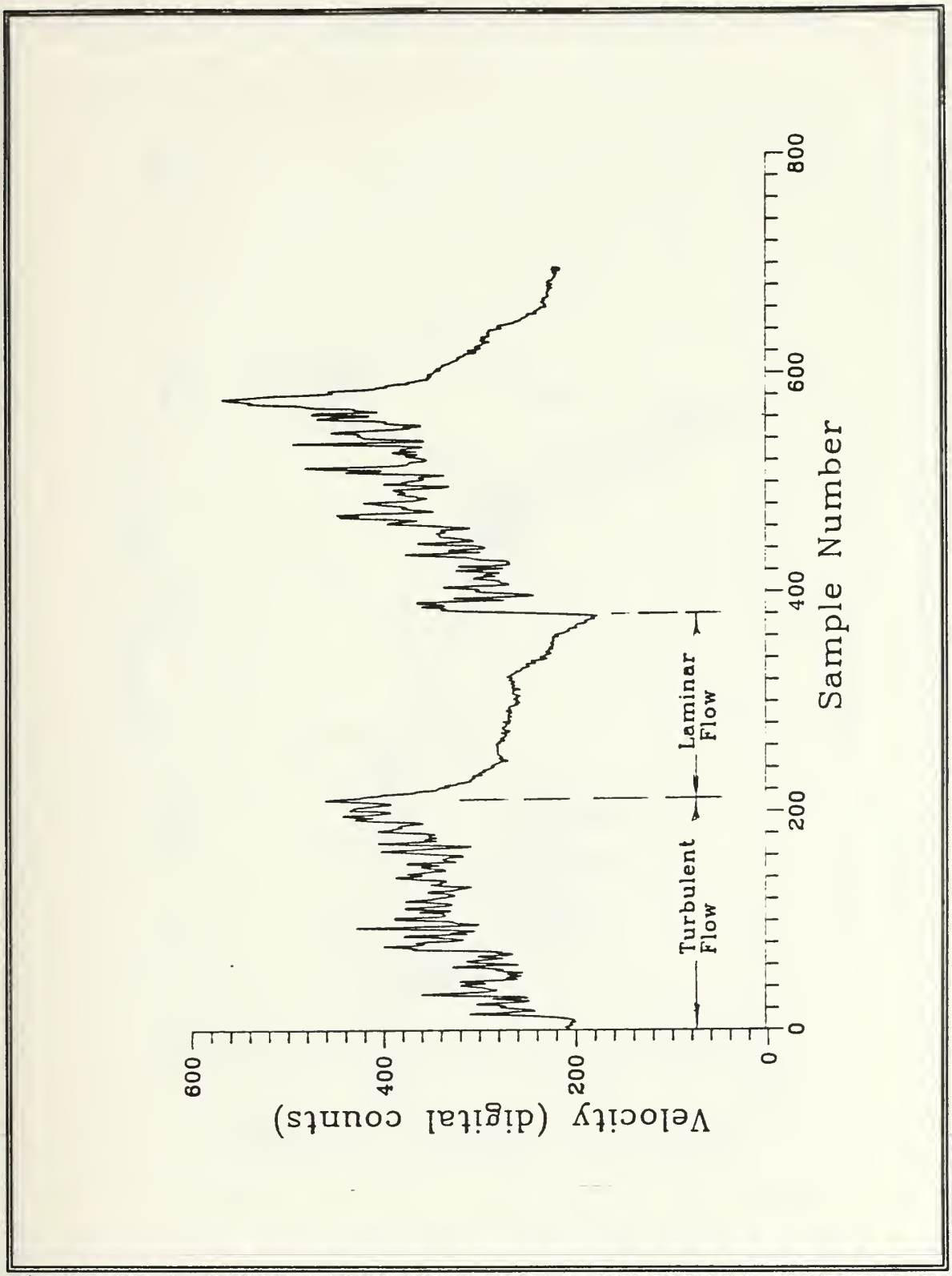

Figure IV-1. PULSE 1 FROM TEST CASE 1 (AN EXAMPLE ENSEMBLE OF 41 PROPELLER PULSES). SAMPLE RATE = 15,000 HZ.

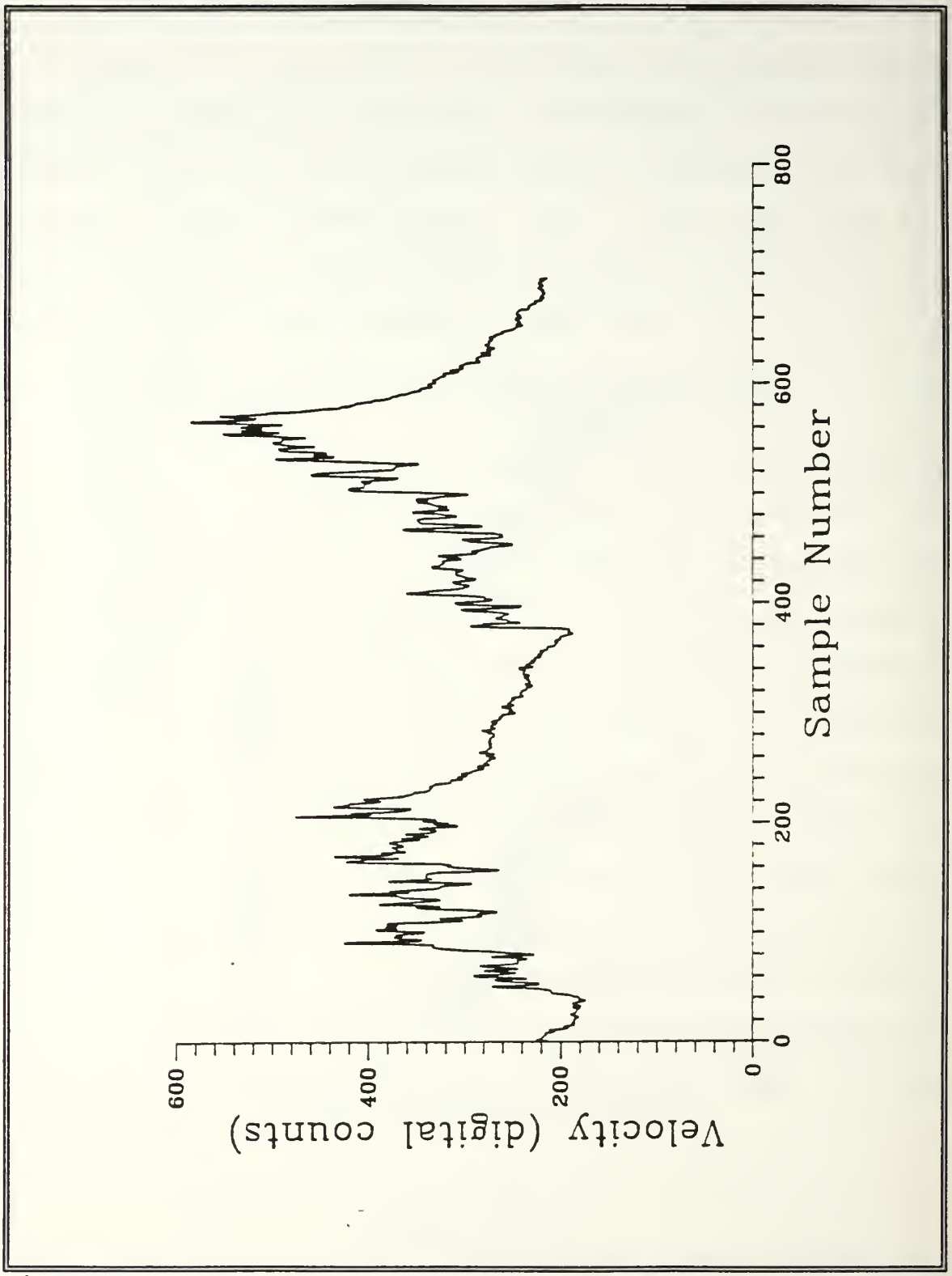

Figure IV-2. PULSE 2 FROM TEST CASE 1 (AN EXAMPLE ENSEMBLE OF 41 PROPELLER PULSES). SAMPLE RATE = 15,000 HZ.

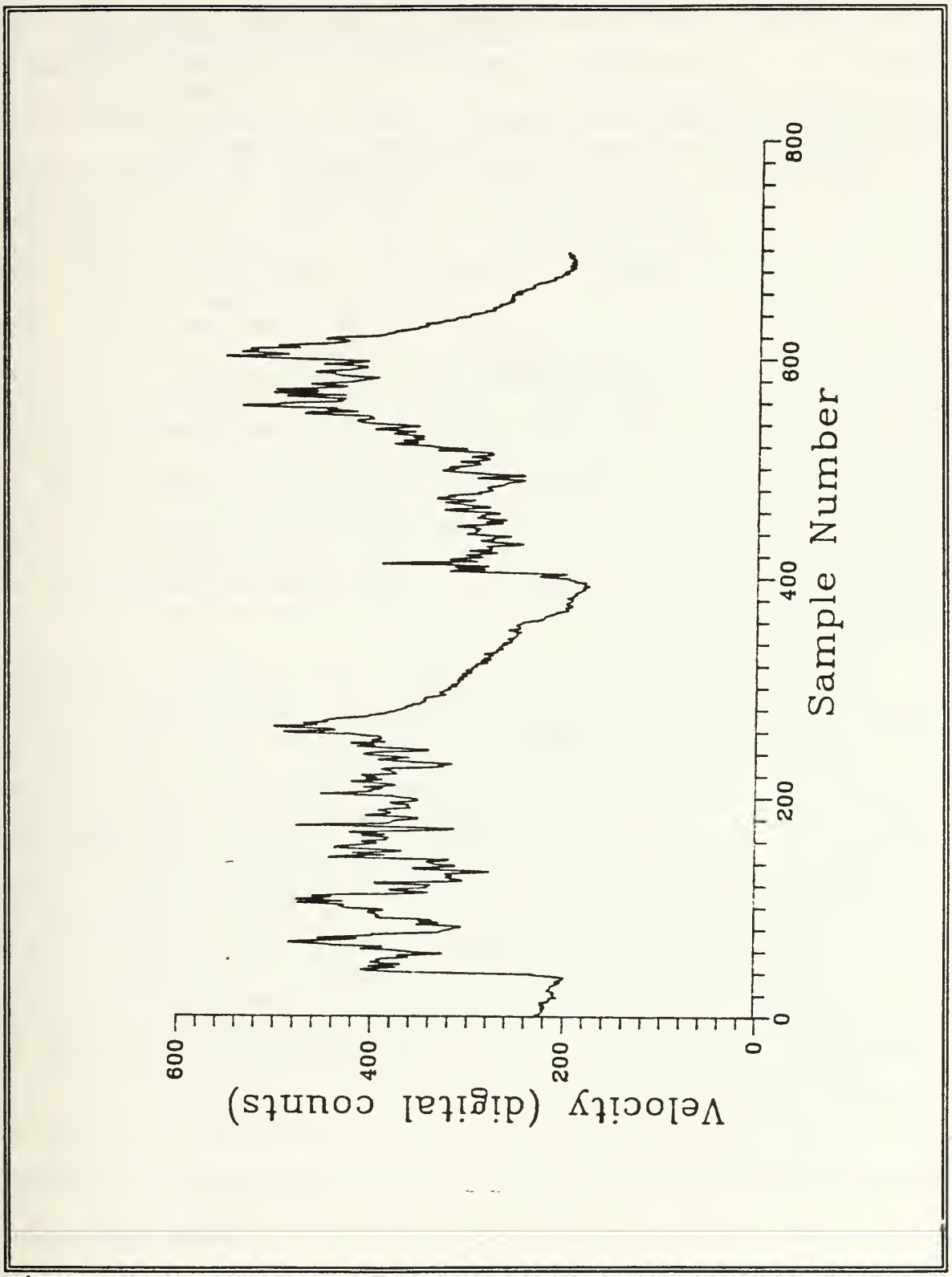

Figure IV-3. PULSE 41 FROM TEST CASE 1 (AN EXAMPLE ENSEMBLE OF 41 PROPELLER PULSES). SAMPLE RATE = 15,000 HZ.

The velocity data were sampled at 15,000 Hz. The two-blade propeller was rotating at approximately 1200 rpm such that a turbulent disturbance (propeller wake) occurs at about 40 Hz. For simplicity, the velocity is presented in digital counts rather than physical units. The data are plotted with straight line segments between each of the data points. The free stream (aircraft) velocity was 65 miles per hour. Time is presented as sample number with sample period $t = 6.666 \times 10^{-5}$ seconds. Each pulse contains 696 time samples in this example. No two-component data (individual measurements of u and v) were acquired during the tests of Reference 2. When needed for software check-out, two-component data will be simulated.

The standard input file format for single-wire velocity data consists of a sequential file containing alternating values of velocity and propeller trigger, starting with velocity. The propeller trigger is a signal generated with a magnetic pickup from one of the propeller blades. Each rotation of the blade generates a sharp voltage spike which is recorded and used to identify the beginning of each propeller pulse. Time is not recorded explicitly but is implied from the sample rate.

B. MEAN VELOCITY DETERMINATION

1. Overview

Equation II-1 describes a common empirical model of fluid velocity, u(t), in a turbulent fluid flow as the sum of the mean velocity, u, and the (zero mean) stationary random fluctuating velocity component, u'. In the boundary layer of an airfoil immersed in a propeller slipstream, the non-stationary mean velocity varies with time as does the high frequency fluctuating component (i.e. the instantaneous velocity is non-stationary in mean). The total instantaneous velocity is actually measured. So the problem is to extract the time varying mean velocity from the measured instantaneous velocity. The classical method for determining the mean of non-stationary data is ensemble averaging. Ensemble averaging works well if a sufficient number of sample records are obtained. For this specific problem, the mean velocity varies periodically with the propeller rotation at a known constant low frequency relative to the turbulent fluctuations. This "low frequency non-stationary mean" empirical model of the velocity data provides an alternate method of analyzing the non-stationary data through low pass filtering [Ref. 6]. Figures IV-1 through IV-3 illustrate the "low frequency mean" nature of the data and justify the use of a low frequency mean empirical model.

Several different algorithms to determine the time varying mean velocity will be examined in this section. The algorithms are based on the concept of ensemble averaging, the low frequency mean empirical model, or a combination of the two.

2. Ensemble Averaging Algorithm

a. Description of the Algorithm

The ensemble averaging algorithm is a straight forward implementation of equation III-8. The ensemble average is computed using the following steps:

- Identify the input and output data file names.
- Separate the input time history file into an ensemble of individual pulses. An array (ENSMBL) is used to store up to 50 propeller pulses of up to 2048 samples each. Read the input data file until the first propeller trigger is found, indicating the beginning of a pulse. Moving down the first column, read each velocity sample into the array. When the next propeller trigger is encountered, begin at the top of the next column and continue reading the velocity samples into the array. Repeat the process until all pulses have been put into the array. Figure IV-4 illustrates how the velocity data are stored into the ensemble array. In the event that each propeller pulse does not contain the same number of data points, the columns are truncated to the length of the shortest column.
- A row in array ENSMBL represents data at the same fixed time point from all of the pulses as illustrated in Figure III-1. Per equation III-3, compute the ensemble average by first summing the velocity values across each row, then dividing each sum by the number of pulses. Store the average of each row in the column array AVG. AVG represents the ensemble average of all the pulses at each time sample.
- Write the ensemble average array AVG to the output file.

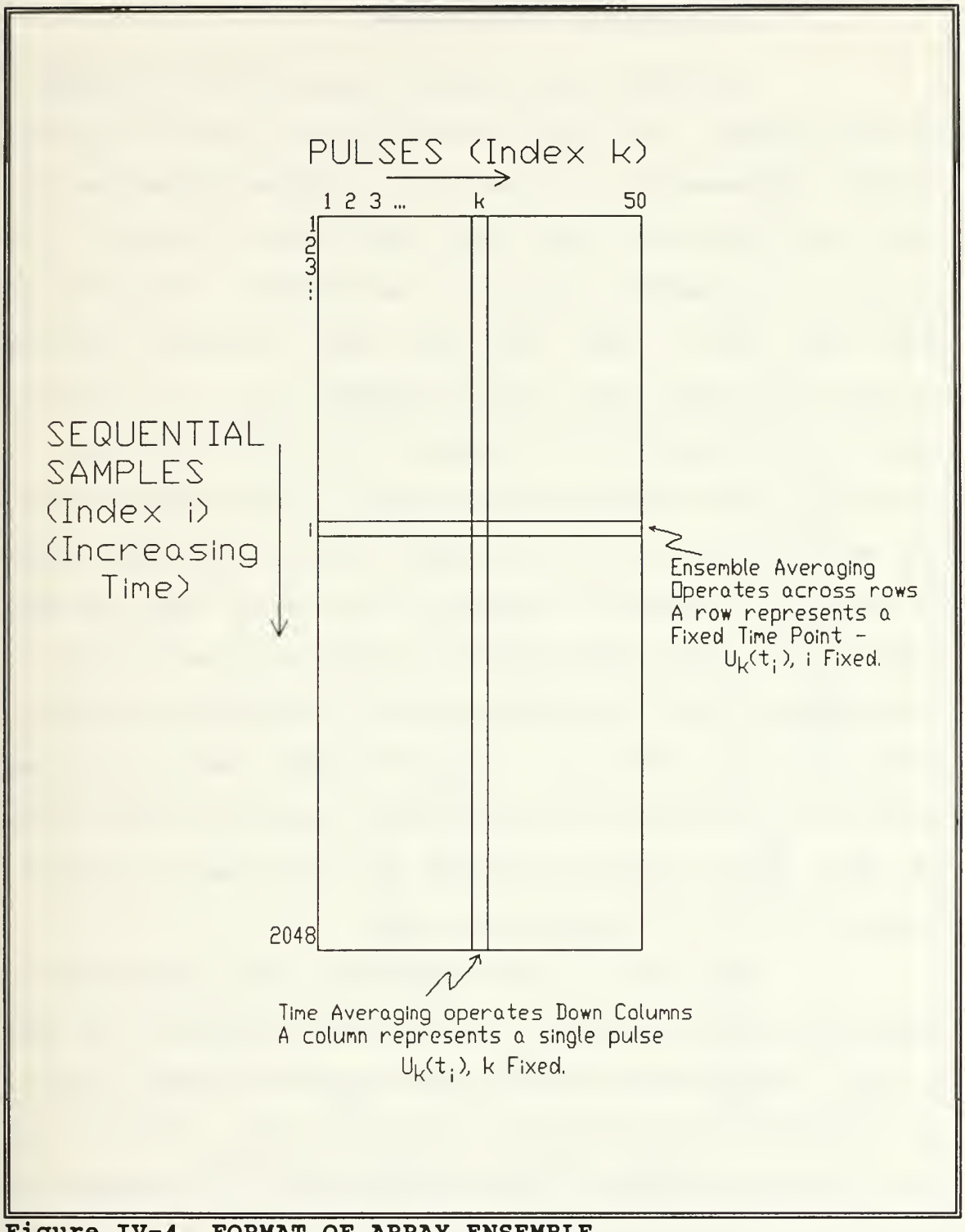

Figure IV-4. FORMAT OF ARRAY ENSEMBLE.

b. Implementation and Examples

The above algorithm was implemented in FORTRAN as program ENSEMBL. The source code and user's guide for program ENSEMBL is presented in Appendix B. Program ENSEMBL was run with the test case input file described in section IV.A. Figure IV-5 presents a plot of the ensemble average of the test case data. Note that the high frequency turbulent fluctuations have been greatly reduced but not eliminated compared to Figures IV-1 through IV-3. Qualitatively, the remaining high frequency fluctuations in the ensemble average may be an indication that 41 pulses is not a sufficient number to obtain a reasonable estimate of the mean using ensemble averaging. This observation provides the motivation for investigation into alternate methods of determining the non-stationary mean velocity. Computer memory and data storage limitations preclude any substantial increase in the volume of data, thus alternate methods are required to obtain a better estimate of the mean velocity.

The part of the algorithm which separates the input file into an ensemble of individual pulses may introduce a small timing error when the length of each pulse is truncated to the length of the shortest pulse. For test case 1, an average of three to four samples (out of 700 samples per pulse) were truncated from the end of each pulse. This timing error is less than one per cent and is considered negligable.

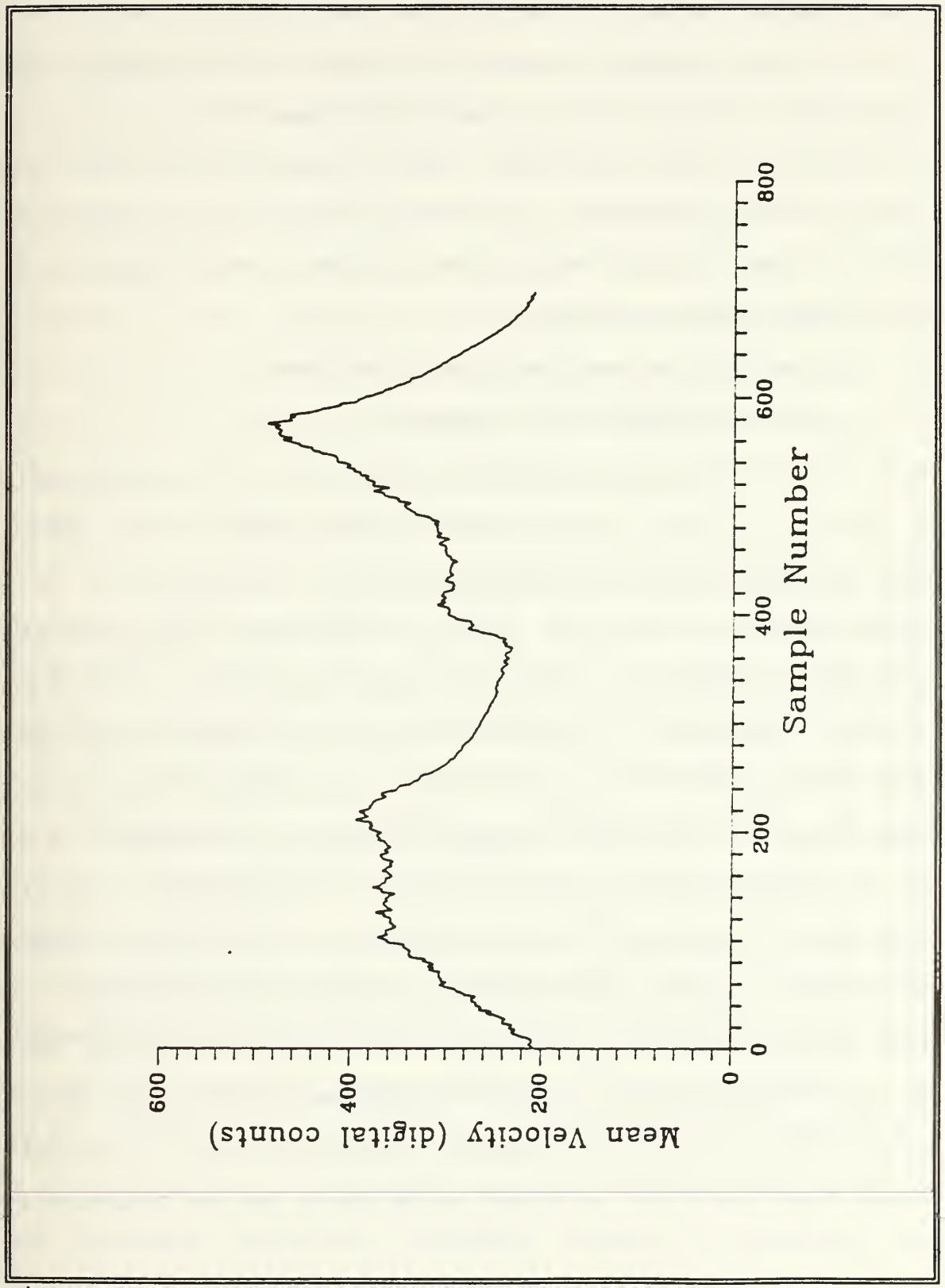

Figure IV-5. ENSEMBLE AVERAGE OF TEST CASE 1.

The error is primarily due to the inability to control the external test conditions, such as airspeed and propeller RPM during the test run. The data of test case 1 were collected in an airborne environment where controlling the test conditions is difficult. Future tests will be conducted in a wind tunnel which will allow the test conditions to be controlled more precisely.

3. Moving Average Smoothing Algorithm

a. Description of the Algorithm

The moving average smoothing algorithm represents a variation on the method used by Howard [Ref. 2] to reduce the unsteady turbulence data presented in Reference 2. This data analysis technique involves applying simple moving average smoothing to the time history data prior to performing ensemble averaging. A moving average is a time average over a short segment or "window" of the time history record. The time average of the short segment becomes the "smoothed" value of the velocity at the center point of the segment. Thus the length of the segment,L, is usually specified as an odd number of points. As the moving average window is moved across the time history record(of length N, where N is much greater than L), a smoothed velocity value is computed for each new center point of the window. If the moving average window is L points long, then the first smoothed value which can be computed is

the (L/2 + 1)'th point. The first L/2 points (and the last L/2 points) in the time history record are set to zero.

The assumption associated with the moving average is that the time history data are stationary over the short window. An empirical model was proposed above for this unsteady turbulence data which supposes that the mean is time varying at a low frequency relative to the random fluctuating velocity component (i.e., the high frequency turbulence). As long as this empirical model is a reasonable description of the actual data, then the assumption of stationarity over short time segments should be valid.

Moving average smoothing is often thought of as a crude low pass digital filter [Ref. 9]. The smoothing operation filters out the high frequency turbulent fluctuations leaving a low frequency "local" mean. As the length of the moving average window is increased (increasing L), lower and lower frequencies are filtered out. So, the resulting mean velocity based on the ensemble average of the smoothed velocity is partly dependent on the frequency response characteristics of the smoothing filter. The problem becomes one of choosing an appropriate smoothing window size to obtain a specific low pass filter cutoff frequency. It will be left up to the aerodynamicist to choose the appropriate cutoff frequency based on knowledge of the nature of unsteady turbulent boundary layers. However, the

frequencies of interest may be bounded. Since the propeller is generating turbulent pulses at a frequency of approximately 40 Hz, the cutoff frequency should be greater than 40 Hz. Landrum and Macha [Ref. 14] show that boundary layers in transition from laminar to turbulent flow contain spectral peaks at approximately 800 Hz. So, the cutoff frequency of a low pass filter should be less than 800 Hz to filter out the fluctuating component. Once a cutoff frequency is selected, all subsequent analysis should be accomplished using only the selected value.

Table IV-1 presents the relationship between the size of the moving average window and the cutoff frequency of the moving average digital filter for a sample rate of 15,000 and 30,000 Hz. The cutoff frequency is defined as the frequency at which the magnitude of the filtered signal has been attenuated by -3 dB (a factor of 0.707) relative to the unfiltered signal. Included in Appendix A is a derivation and discussion of the frequency response characteristics of the moving average digital filter. As detailed in Appendix A, the moving average smoothing algorithm possesses poor digital filter characteristics including a severe Gibb's phenomenon (positive and negative ripples in the frequency response function, see Appendix A).

Table IV-1. APPROXIMATE CUTOFF FREQUENCIES FOR MOVING AVERAGE WINDOWS

Window Size	Cutoff Freq.(Hz) (Sample rate=15000 Hz)	Cutoff Freq.(Hz) (Sample rate=30000 Hz)
3	2330	4660
5	1355	2710
7	960	1920
9	745	1485
11	610	1220
13	515	1030
15	445	890
17	395	790
19	355	710
21	320	640
23	290	580
25	270	540
27	250	500
29	230	460
31	215	430
33	205	410
35	190	380
37	180	360
39	175	350

The moving average algorithm for determining the non-stationary mean velocity is accomplished through the following steps:

- Identify the input and output data file names and enter the desired moving average window size (an odd number of points).
- Separate the input time history record into an ensemble of individual pulses. As previously described, each column of array ENSMBL represents a single propeller pulse.
- Apply moving average smoothing to each column (propeller pulse) of array ensemble. Write the smoothed values of velocity back into array ENSMBL for storage.
- Perform ensemble averaging on the smoothed velocity to obtain the final mean velocity.
- Write the output file.

b. Implementation and Examples

The moving average algorithm was implemented as program MOVAVE. The source code and user's guide for program MOVAVE is presented in Appendix B. Using test case 1 as the input file, program MOVAVE produced the results presented in Figure IV-6. A 21 point moving average window was selected yielding a cutoff frequency of 320 Hz. Comparing the moving average results of Figure IV-6 to the pure ensemble average shown in Figure IV-5, observe that, as expected, the small high frequency fluctuations remaining in the ensemble average have been removed. Figure IV-7 presents the results of program MOVAVE with a 7 point smoothing window selected (960 Hz. cutoff frequency). Observe that some high frequency signal is still present in the mean velocity suggesting that a significant amount of turbulent energy exists below 960 Hz. (Later spectral analysis will show this to be true.) Figure IV-8 shows the results of program MOVAVE with a 41 point moving average window (corresponding to a cutoff frequency of 165 Hz.). Note that this example is not significantly different from the 21 point window results shown in Figure IV-6. The similarity of Figures IV-6 and IV-8 suggests that the 21 point window has succeeded in removing most of the high frequency fluctuating velocity component.

Combining ensemble averaging with a "low frequency mean" empirical model (moving average smoothing)

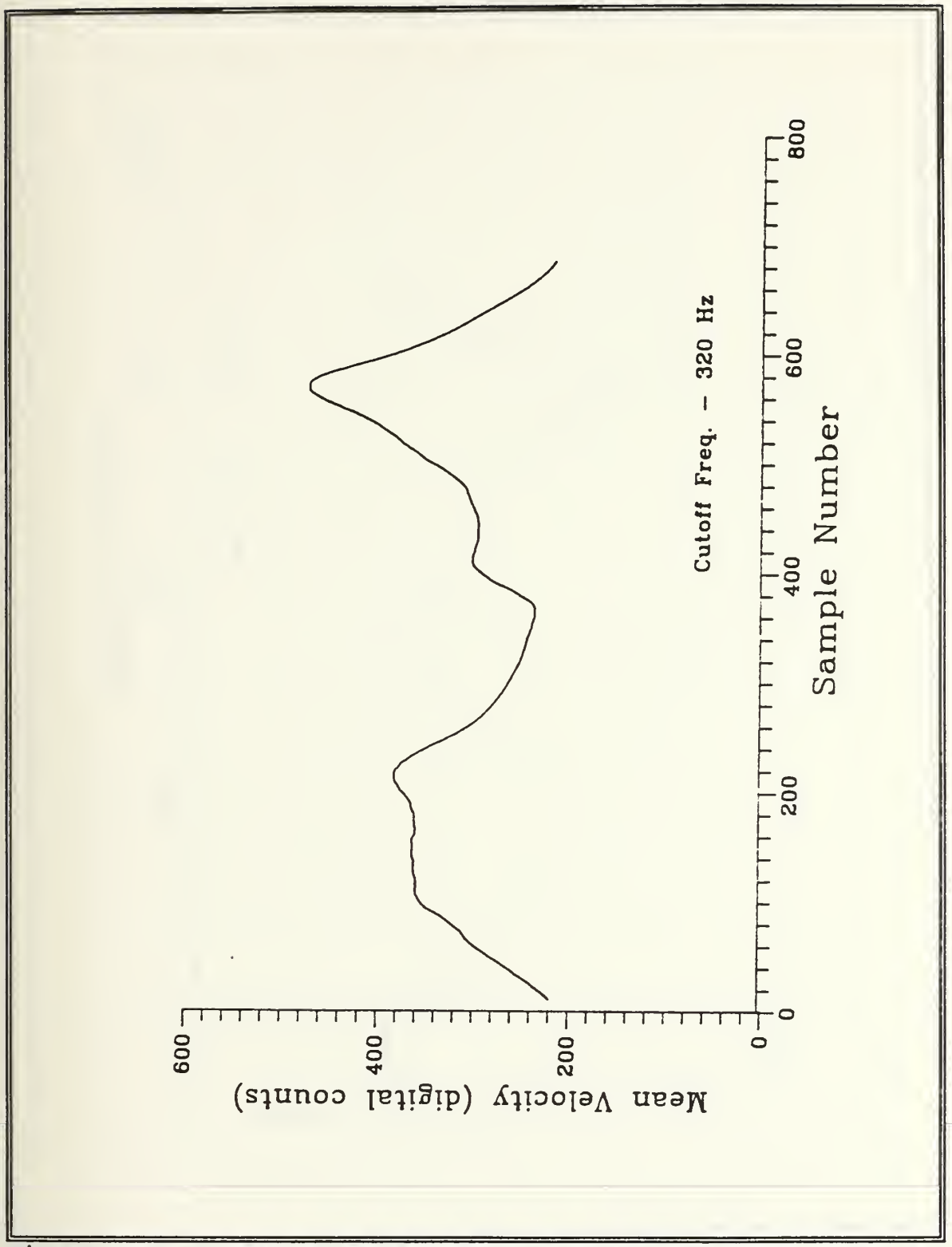

Figure IV-6. RESULT OF PROGRAM MOVAVE ON TEST CASE 1. (SMOOTHED USING A 21 POINT MOVING AVERAGE, THEN ENSEMBLE AVERAGED.)

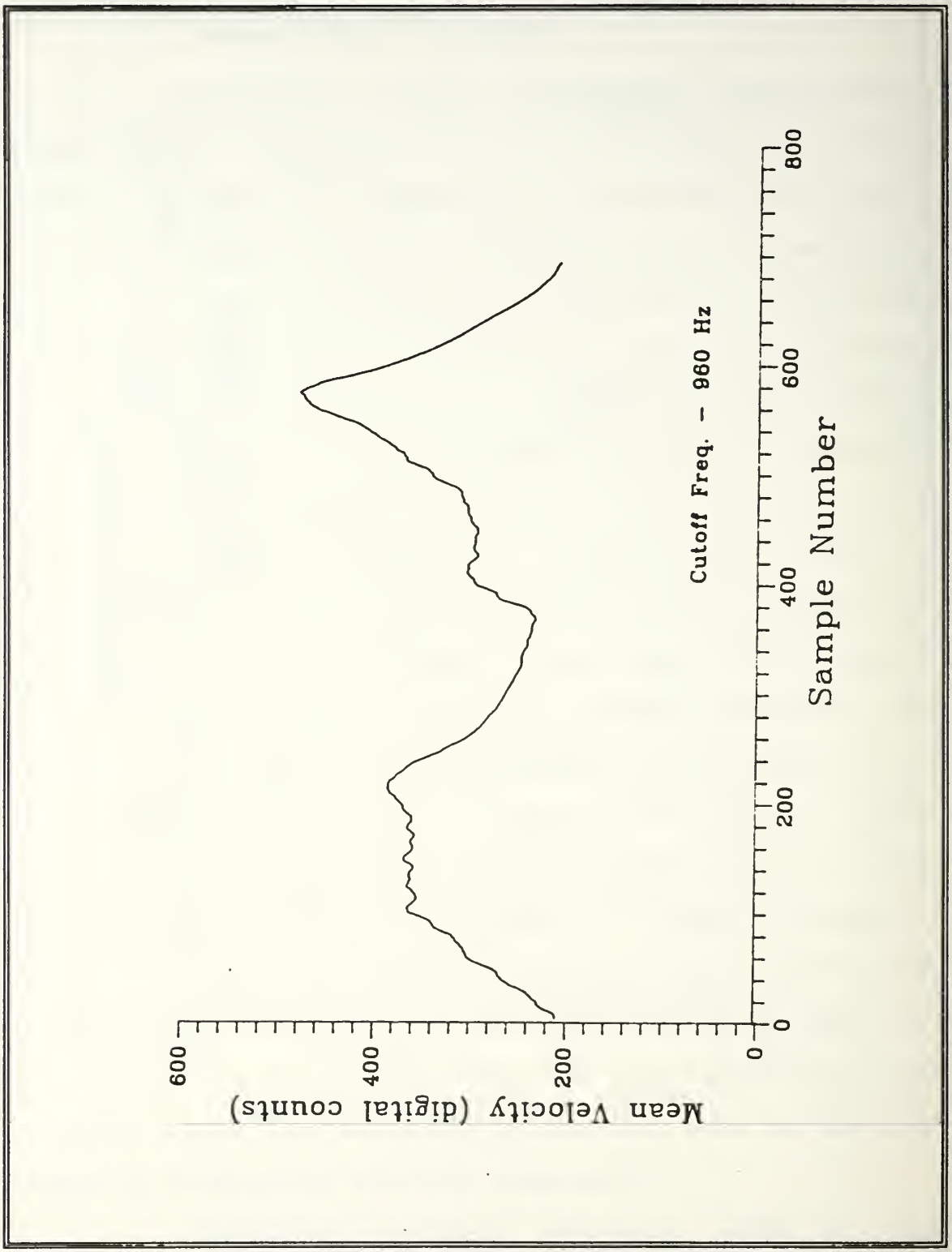

Figure IV-7. RESULT OF PROGRAM MOVAVE ON TEST CASE 1. (SMOOTHED USING A 7 POINT MOVING AVERAGE, THEN ENSEMBLE AVERAGED.)

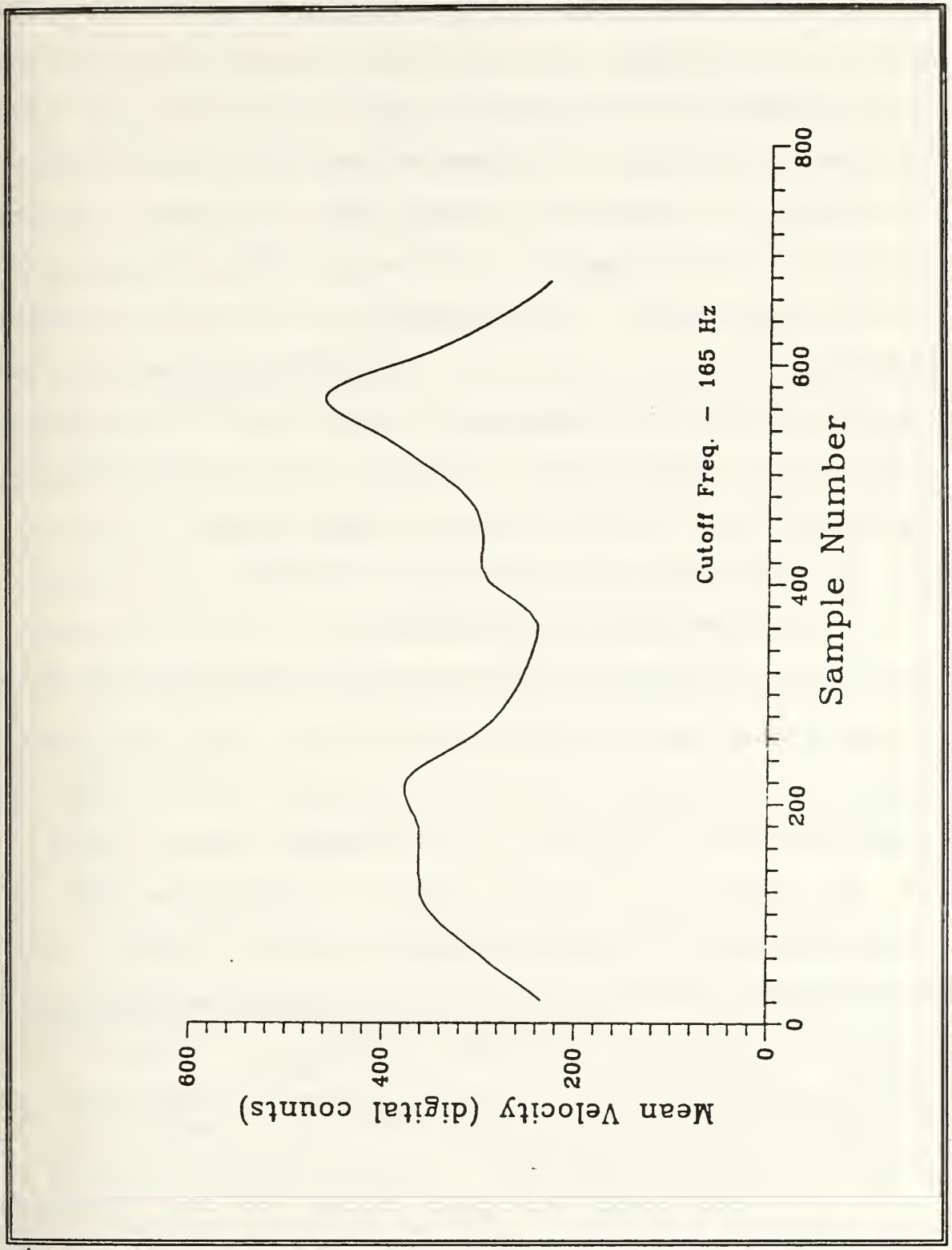

Figure IV-8. RESULT OF PROGRAM MOVAVE ON TEST CASE 1. (SMOOTHED USING A 41 POINT MOVING AVERAGE, THEN ENSEMBLE AVERAGED.)

provides at least a qualitative improvement in the estimate of the non-stationary mean velocity without increasing the volume of data. Since the mean of the non-stationary velocity is somewhat arbitrarily defined as the "low frequency part", it becomes difficult to quantify any improvement in the accuracy of the estimate of the mean. The resulting mean velocity is strongly dependent upon the size of the moving average smoothing window (i.e. the cutoff frequency). The cutoff frequency must be carefully selected to be consistent with the low frequency mean empirical model and the physical nature of the unsteady turbulent boundary layer.

4. Frequency Domain Smoothing Algorithm

a. Description of the Algorithm

Frequency domain smoothing involves the more classical approach to digital filtering. The time history record is transformed into the frequency domain using the discrete Fourier transform. The frequency response function of the desired low pass filter is multiplied by the transformed data to remove the high frequency content. Then the data are transformed back into the time domain as the smoothed (low pass filtered) velocity. Recall that multiplication in the frequency domain is equivalent to convolution in the time domain.

The ideal low pass filter (in the frequency domain) looks like Figure IV-9, where the step occurs at the

desired cutoff frequency (f_c). As discussed in section III.B.6., practical finite length time series cause errors in the Fourier series representations of functions due to truncation of the Fourier coefficients. Therefore the "ideal" filter cannot be achieved practically. Figure IV-10 illustrates the result of the truncated Fourier series representation of the ideal filter. The ripples in the frequency response function are known as the Gibb's phenomenon [Ref. 9]. The Gibb's phenomenon is an undesirable characteristic for a filter. Smooth, continuous filter frequency response functions are selected to minimize the ripple effect (i.e. to reduce the magnitude of the high frequency Fourier coefficients per section III.B.6.).

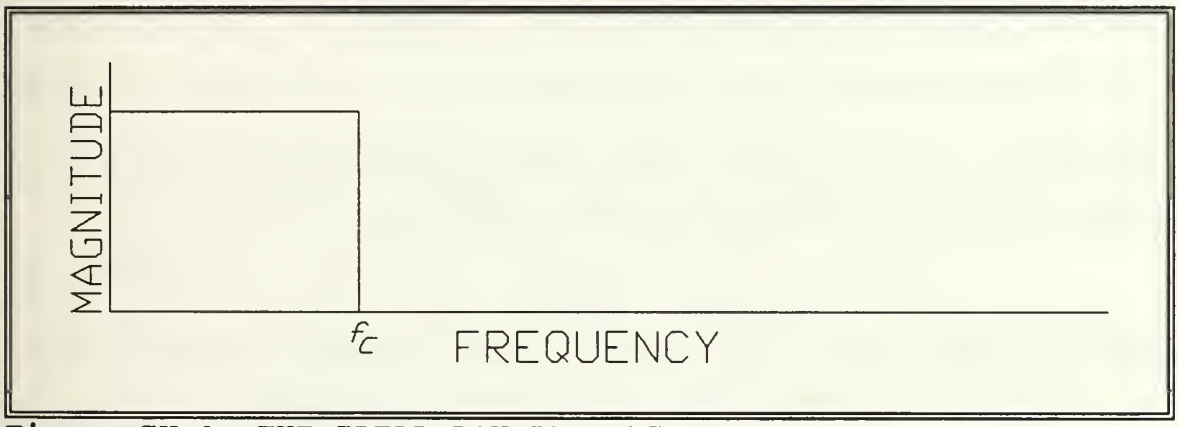

Figure IV-9. THE IDEAL LOW PASS FILTER.

The specific filter frequency response function used in the smoothing algorithm presented here, is a parabolic shaped function. The filter function is one at zero frequency and falls smoothly to zero at a frequency which depends on the user specified smoothing window size (the

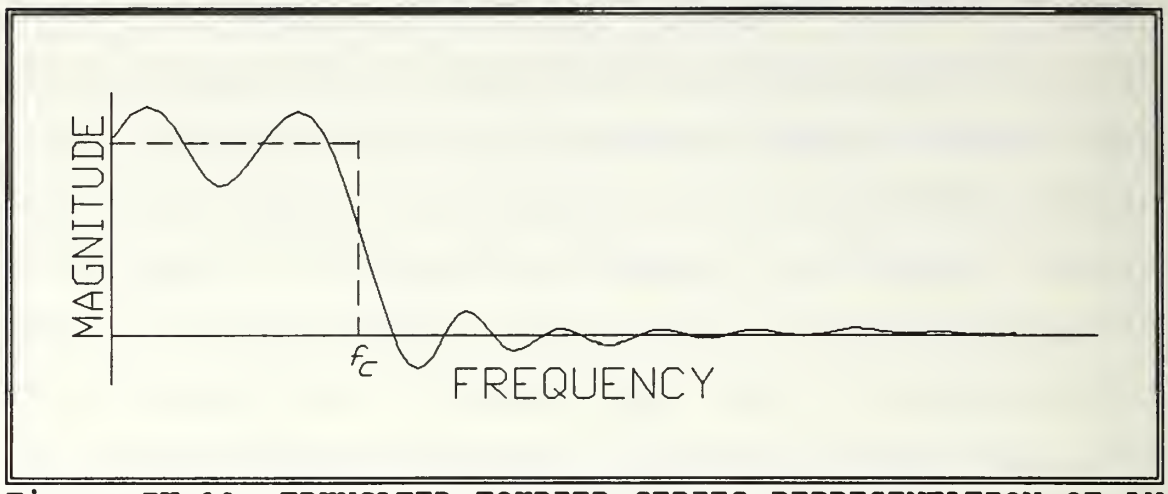

Figure IV-10. TRUNCATED FOURIER SERIES REPRESENTATION OF AN IDEAL FILTER.

cutoff frequency). An analysis of the frequency response characteristics of the parabolic filter is presented in Appendix A. Other frequency domain filters with different frequency response characteristics may be implemented, such as the von Hann window described in section III.B.6. Table IV-2 presents the cutoff frequency, (f_c), as a function of the smoothing window size, (L), the sample rate, (f_s), and the size (number of points) of the DFT, (M).

The frequency domain smoothing algorithm for determining the non-stationary mean velocity is very similar to the moving average algorithm. Frequency domain low pass filter smoothing is accomplished prior to ensemble averaging instead of a moving average. The low pass filter algorithm requires a program to perform the DFT and IDFT. The following steps describe the algorithm:

- Identify the input and output data file names and enter the desired frequency domain smoothing window size.

Table IV-2. APPROXIMATE CUTOFF FREQUENCIES FOR FREQUENCY DOMAIN LOW PASS FILTER.

L	f_c(Hz) (f^s=15000, M=1024)	f^c(Hz) (f^s=30000, M=2048)
2	4060	8120
3	2710	5410
4	2030	4060
5	1625	3250
6	1355	2710
7	1160	2320
8	1015	2030
9	900	1805
10	810	1625
11	740	1475
12	675	1350
13	625	1250
14	580	1160
15	540	1080
20	405	810
25	325	650
30	270	540
35	230	465
40	205	405
45	180	360
50	165	325

- Separate the input time history record into an ensemble of individual pulses. As previously described, each column of array ENSMBL represents a single propeller pulse.

- Apply the low pass filter smoothing to each column (propeller pulse) of array ensemble. This step requires an FFT routine to accomplish the DFT and IDFT. Write the smoothed values of velocity back into array ENSMBL for storage. The smoothing subroutine and the FFT subroutine were extracted from Press, et al. [Ref. 12].

- Perform ensemble averaging on the smoothed velocity to obtain the final mean velocity.

- Write the output file.

b. Implementation and Examples

The frequency domain smoothing algorithm was implemented as program SMUMEAN. The source code and user's guide for program SMUMEAN is included in Appendix B. The low pass filter and FFT subroutines used in this implementation are from Press, et al. [Ref. 12]. Figure IV-11 presents the results of applying program SMUMEAN on test case 1 for a 25 point smoothing window (a cutoff frequency of 325 Hz). Figures IV-12 and IV-13 present the results of program SMUMEAN for an 8 point (f_c = 1015 Hz) and a 51 point (f_c = 160 Hz) smoothing window respectively. These cutoff frequencies were chosen as close as possible to those presented in Figures IV-6 through IV-8.

Comparisons of the frequency domain smoothing results with the moving average smoothing results indicate negligible differences in the corresponding mean velocity between the two algorithms for similar cutoff frequencies. These results are consistent with the fact that the frequency response function of the frequency domain smoothing algorithm is within 1% of the frequency response function of the moving average algorithm at frequencies between zero and the cutoff frequency (see Appendix A). The frequency response functions of the two methods begin to diverge significantly above the cutoff frequency. But at this point, the magnitude of the high frequency fluctuating velocity component has been

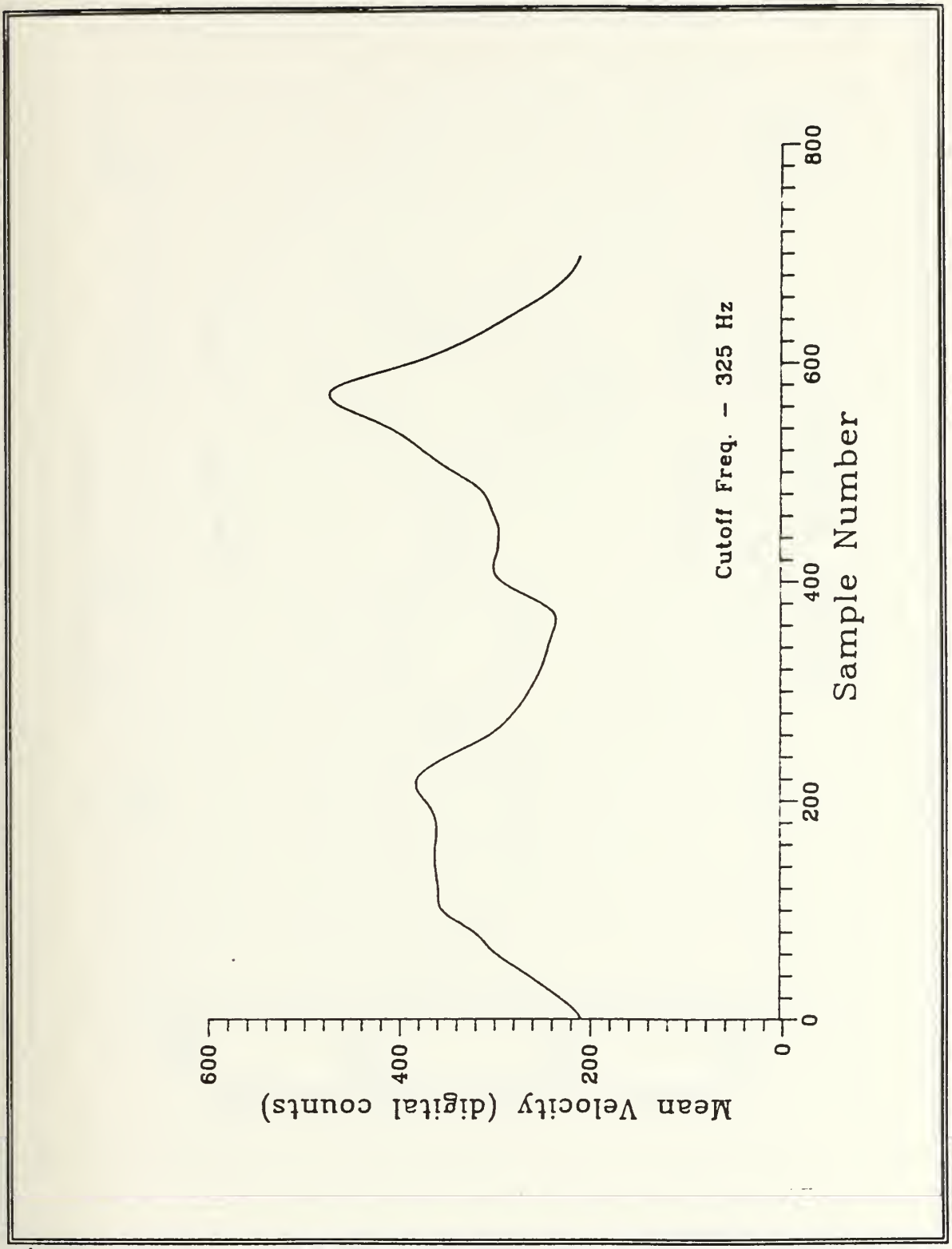

Figure IV-11. RESULT OF PROGRAM SMUMEAN ON TEST CASE 1. (SMOOTHED USING A 25 POINT LOW PASS FILTER, THEN ENSEMBLE AVERAGED.)

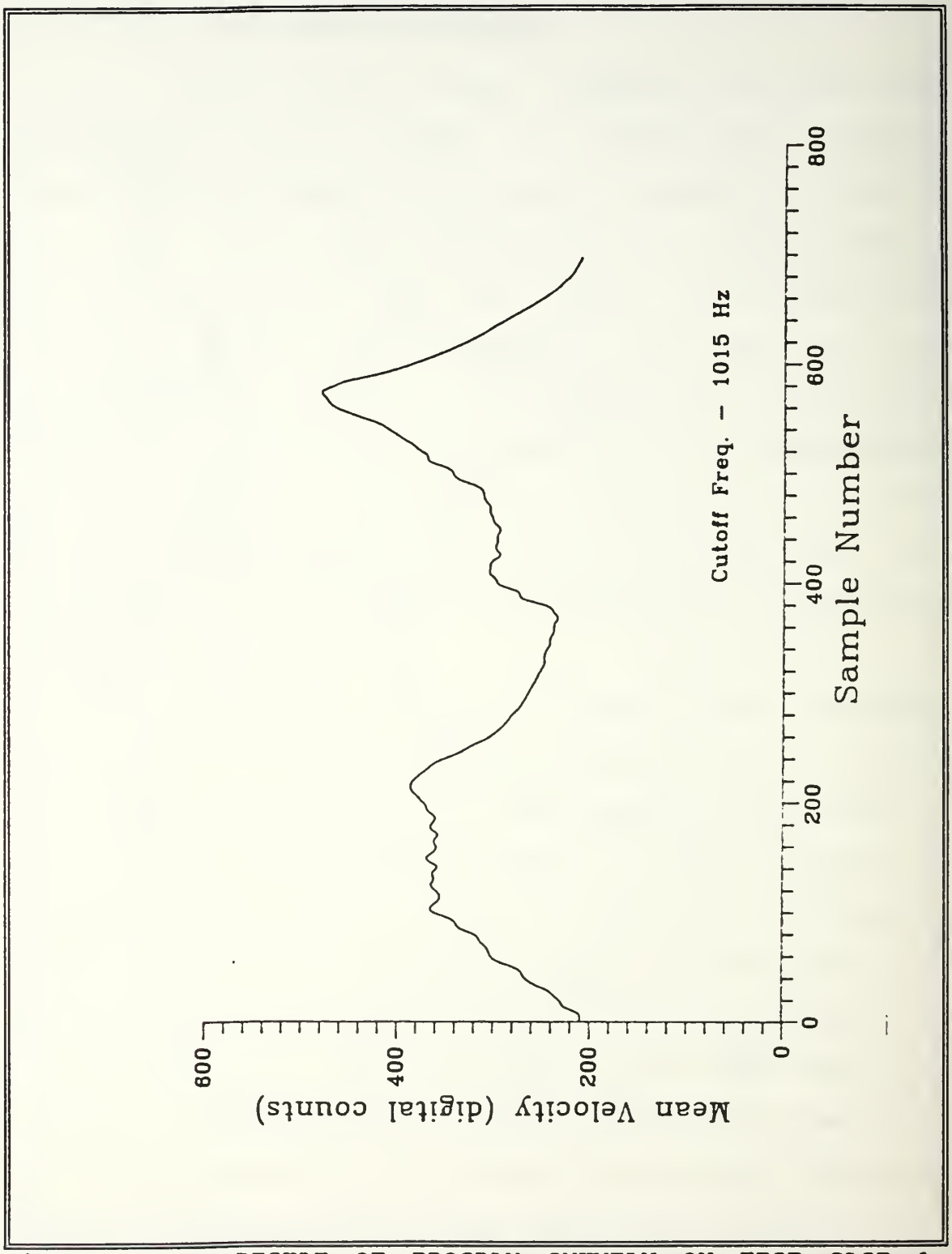

Figure IV-12. RESULT OF PROGRAM SMUMEAN ON TEST CASE 1. (SMOOTHED USING AN 8 POINT LOW PASS FILTER, THEN ENSEMBLE AVERAGED.)

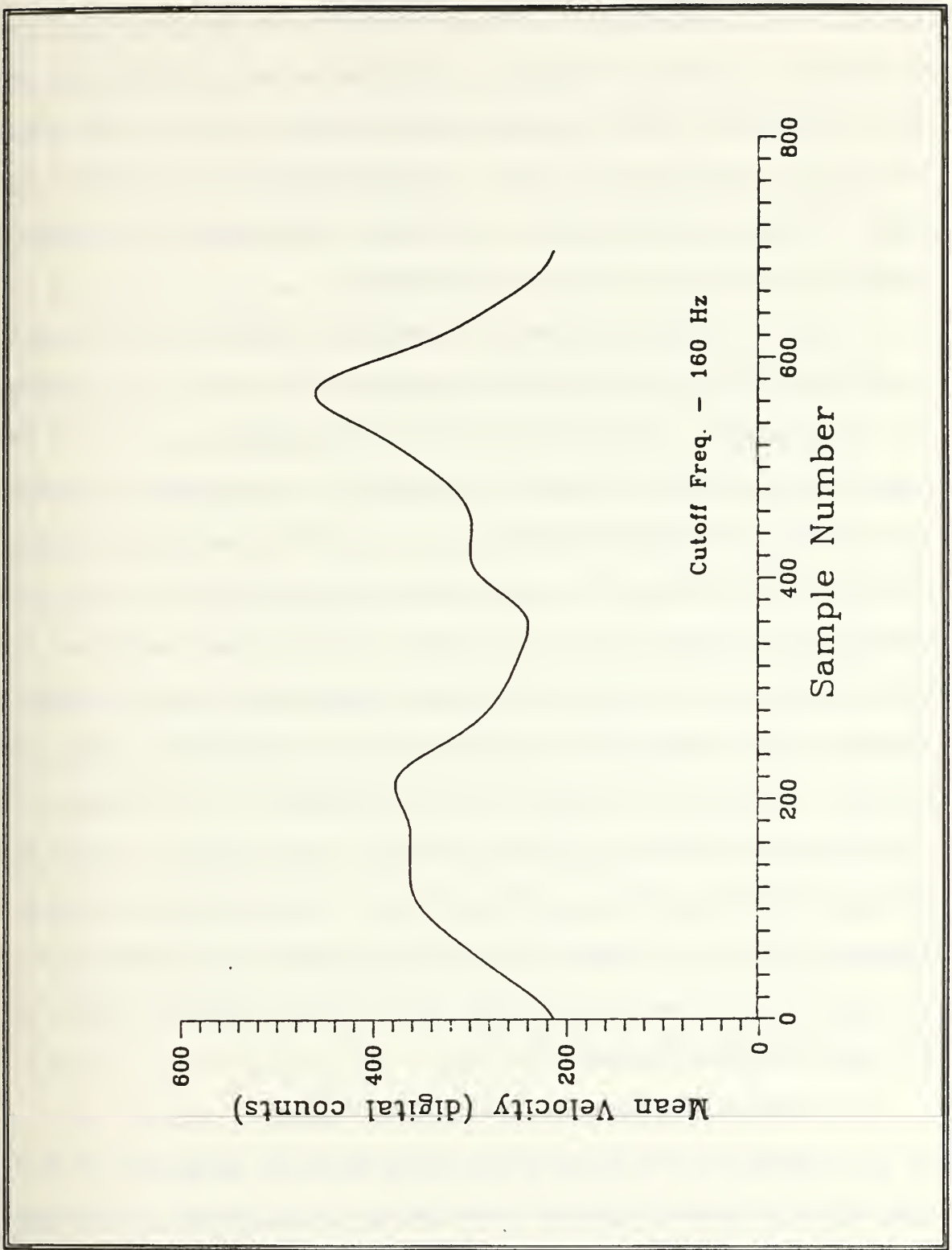

Figure IV-13. RESULT OF PROGRAM SMUMEAN ON TEST CASE 1. (SMOOTHED USING A 51 POINT LOW PASS FILTER, THEN ENSEMBLE AVERAGED.)

significantly attenuated such that there is little effect on the resulting mean velocity. Choosing a low pass filter for the frequency domain smoothing algorithm with a frequency response function that better approximates the "ideal" low pass filter would cause a greater (though still minor) difference between the two algorithms.

Combining ensemble averaging with frequency domain smoothing (low pass filtering) yields similar results compared to the moving average smoothing algorithm. A similar improvement is obtained in the estimate of the mean of a non-stationary velocity measurement as was obtained with program MOVAVE, when compared to ensemble averaging alone. Again, the cutoff frequency must be chosen with care because the resulting mean velocity is strongly dependent upon the cutoff frequency selected. The frequency domain smoothing algorithm has the advantage of increased flexibility in the choice of the low pass filter characteristics. Furthermore, frequency domain smoothing has a larger choice of cutoff frequencies (compare Table IV-1 with Table IV-2), and the beginning and end points are not lost (set to zero) as occurs using the moving average algorithm.

5. Ensemble Averaging Before Smoothing

Both of the algorithms described in sections IV.B.3. and IV.B.4 above employ smoothing (low pass filtering) techniques prior to ensemble averaging. Both moving average

smoothing and frequency domain smoothing are linear operations, as is ensemble averaging. Therefore, identical mean velocity results are obtained when the ensemble average operation is performed prior to the smoothing operation. However, a substantial increase in computational efficiency is achieved because the smoothing operation is only performed on a single ensemble averaged velocity record instead of on each of the individual pulses which comprise the ensemble.

Program MEANSMU was implemented to demonstrate the increased computational speed achieved by ensemble averaging prior to smoothing. The algorithm for program MEANSMU is the same as program SMUMEAN presented in section IV.B.4. above, except that steps three and four are reversed. Appendix B presents the source code and user's guide for program MEANSMU. The results presented in Figure IV-11 required approximately 143 seconds of computer time for program SMUMEAN. Program MEANSMU produces identical results in only 34 seconds of computer time. (Note that the computer was a ten megahertz IBM PC/AT clone equipped with an 80287 math co-processor). No results are presented for program MEANSMU since they are identical to those shown in Figures IV-11 through IV-13.

6. Summary--Mean Velocity Determination

Program ENSEMBL provides a method for estimating the non-stationary mean velocity of an ensemble of propeller blade wake passages by ensemble averaging. Program MOVAVE and

SMUMEAN provide alternate methods for obtaining improved estimates of the non-stationary mean velocity when the number of pulses (i.e. the number of sample records) in the ensemble is small. Program MOVAVE and SMUMEAN yield similar results. Program SMUMEAN maintains the following advantages over program MOVAVE:

- Increased flexibility in the number of available cutoff frequencies.
- Increased flexibility in the choice of the frequency response characteristics of the low pass filter used for smoothing.
- No loss of data at the beginning and end of the mean velocity data record.

Program MEANSMU possesses these same advantages plus the added advantage of increased computational efficiency. Therefore, program MEANSMU is the recommended method for determining the non-stationary mean velocity. It is further recommended that alternate frequency domain low pass filters be implemented and tested in the "smoothing" algorithm of program MEANSMU to determine if there is any significant difference or improvement in the non-stationary mean velocity estimate.

C. TURBULENCE INTENSITY DETERMINATION

1. Overview

In Chapter II, turbulence intensity was defined as the root mean square of the fluctuating velocity component. For steady turbulence, the fluctuating velocity component is

simply the difference between the instantaneous velocity and the mean velocity. Thus, for steady turbulence, turbulence intensity is very similar to the statistical variance. But for unsteady (non-stationary) turbulence, the fluctuating velocity component is defined as the difference between the instantaneous velocity and the local smoothed velocity (the local mean), not the non-stationary ensemble averaged mean velocity. The local smoothed velocity is obtained using either the moving average smoothing or the frequency domain low pass filter smoothing. And the "mean" operation in the root mean square of the fluctuating component is the ensemble average. This definition prevents variations in the local mean from pulse to pulse to be incorrectly included in the turbulence intensity measure. In other words, for unsteady turbulence, turbulence intensity measures only the variation due to the fluctuating velocity component relative to the smoothed velocity (local mean). But statistical variance measures the total variation of the instantaneous velocity due to variations in both the fluctuating velocity component (turbulence) and variations in the local smoothed velocity. Figure IV-14 presents an illustration.

Using this "local mean" definition of turbulence intensity, two algorithms have been developed using the two smoothing algorithms which were presented in section IV.A.

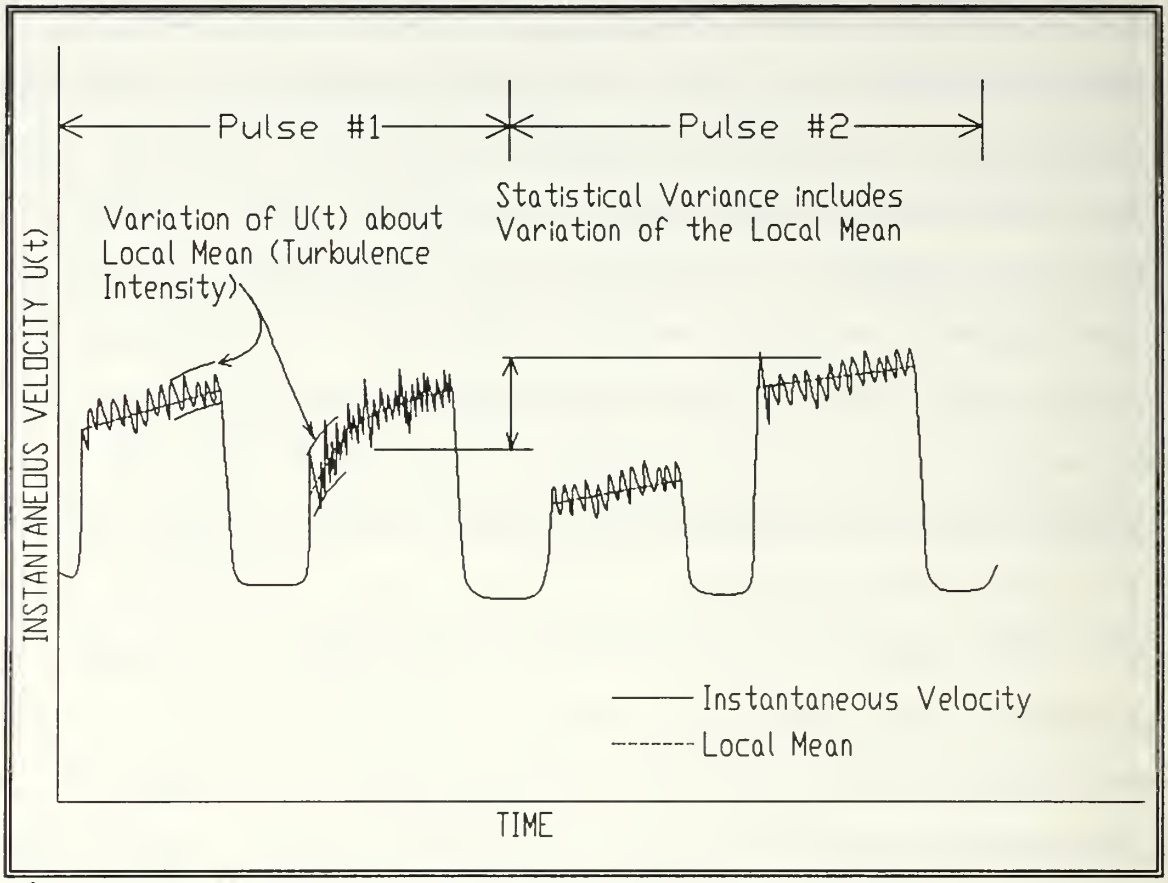

Figure IV-14. ILLUSTRATION OF TURBULENCE INTENSITY (BASED ON "LOCAL MEAN") VERSUS STATISTICAL VARIANCE (BASED ON ENSEMBLE AVERAGE OF LOCAL MEAN)

above. The turbulence intensity algorithms are presented in the following paragraphs.

2. Turbulence Intensity from Moving Average Smoothing

a. Description of the Algorithm

Moving average smoothing was described in section IV.A.3. above. The local mean (smoothed velocity) at a given instant in time, $u(t_i)_{LOCAL}$, is used to determine the local fluctuating velocity component. Expanding on equation II-3, turbulence intensity is computed as:

$$\text{TURBULENCE INTENSITY} = \frac{\sqrt{\frac{1}{N}\sum_{k=1}^{N}\left[u(t_1)-\bar{u}(t_1)_{LOCAL}\right]^2}}{U_{edge}} \qquad \text{(IV-1)}$$

where k is the ensemble index, N is the total number of propeller pulses, and U_{edge} is a reference velocity (usually the mean velocity at the edge of the boundary layer) used to normalize the turbulence intensity. U_{edge} is constant for a particular boundary layer profile.

The assumptions which apply to the moving average mean velocity algorithm also apply to the moving average turbulence intensity algorithm. The instantaneous velocity data is assumed stationary over the short smoothing window such that time averaging over the short segment may be used to estimate the local mean. Also the data are assumed to behave according to the "low frequency mean" empirical model. Therefore, as with the moving average mean velocity algorithm, the turbulence intensity results are a strong function of cutoff frequency (smoothing window size). Table IV-1 presents the relationship between moving average window size and cutoff frequency. As before, the choice of an appropriate cutoff frequency will be left up to the aerodynamicist. The frequency response characteristics of moving average smoothing is presented in Appendix A. For consistent results, the same

cutoff frequency should be used in the turbulence intensity analysis as was used in the mean velocity analysis.

The moving average turbulence intensity algorithm is accomplished through the following steps:

- Identify the input and output data file names and enter the desired moving average window size (an odd number of points).
- Separate the input time history record into an ensemble of individual pulses. As previously described, each column of array ENSMBL represents a single propeller pulse.
- Apply moving average smoothing to each column (propeller pulse) of array ensemble. Subtract the instantaneous velocity from the smoothed velocity to get the fluctuating velocity component. Square the fluctuating velocity component and write the result back into array ENSMBL for storage.
- Perform ensemble averaging on the squared fluctuating velocity to obtain the mean square fluctuating velocity. Then compute the square root and normalize (divide by U_{edge}) to obtain the turbulence intensity.
- Write the output file.

b. Implementation and Examples

The moving average turbulence intensity algorithm was implemented as program TURINTMA. The source code and user's guide for program TURINTMA is presented in Appendix B. Figures IV-15, IV-16, and IV-17 present the results of program TURINTMA (using test case 1) for cutoff frequencies of 320 Hz, 960 Hz and 165 Hz respectively. These cutoff frequencies correspond to moving average window sizes of 21 points, 7 points and 41 points respectively. The turbulence intensity

results presented in Figures IV-15, IV-16 and IV-17 correspond to the mean velocity results presented in Figures IV-6, IV-7 and IV-8. These data show the time dependent, periodic nature of the unsteady turbulent boundary layer as the fluid flow oscillates between the turbulent and laminar states.

In comparing the turbulence intensities of Figures IV-15, IV-16 and IV-17, observe that as cutoff frequency is decreased, turbulence intensity level increases. This illustrates the strong relationship between cutoff frequency and turbulence intensity. The cutoff frequency must be carefully selected to be consistent with the low frequency mean empirical model and the physical nature of the unsteady turbulent boundary layer. Note that the moving average smoothing has caused the loss of a small segment of data at the beginning and end of the data record.

3. Turbulence Intensity from Frequency Domain Smoothing

a. Description of Algorithm

Frequency domain low pass filter smoothing was described in section IV.A.4. above. The turbulence intensity algorithm using frequency domain smoothing is essentially the same as that using moving average smoothing. The key difference is the use of low pass frequency domain filtering to obtain the smoothed (local mean) velocity, $u(t_i)_{LOCAL}$. The turbulence intensity is still computed using equation IV-1. Table IV-2 presents the relation between the smoothing window

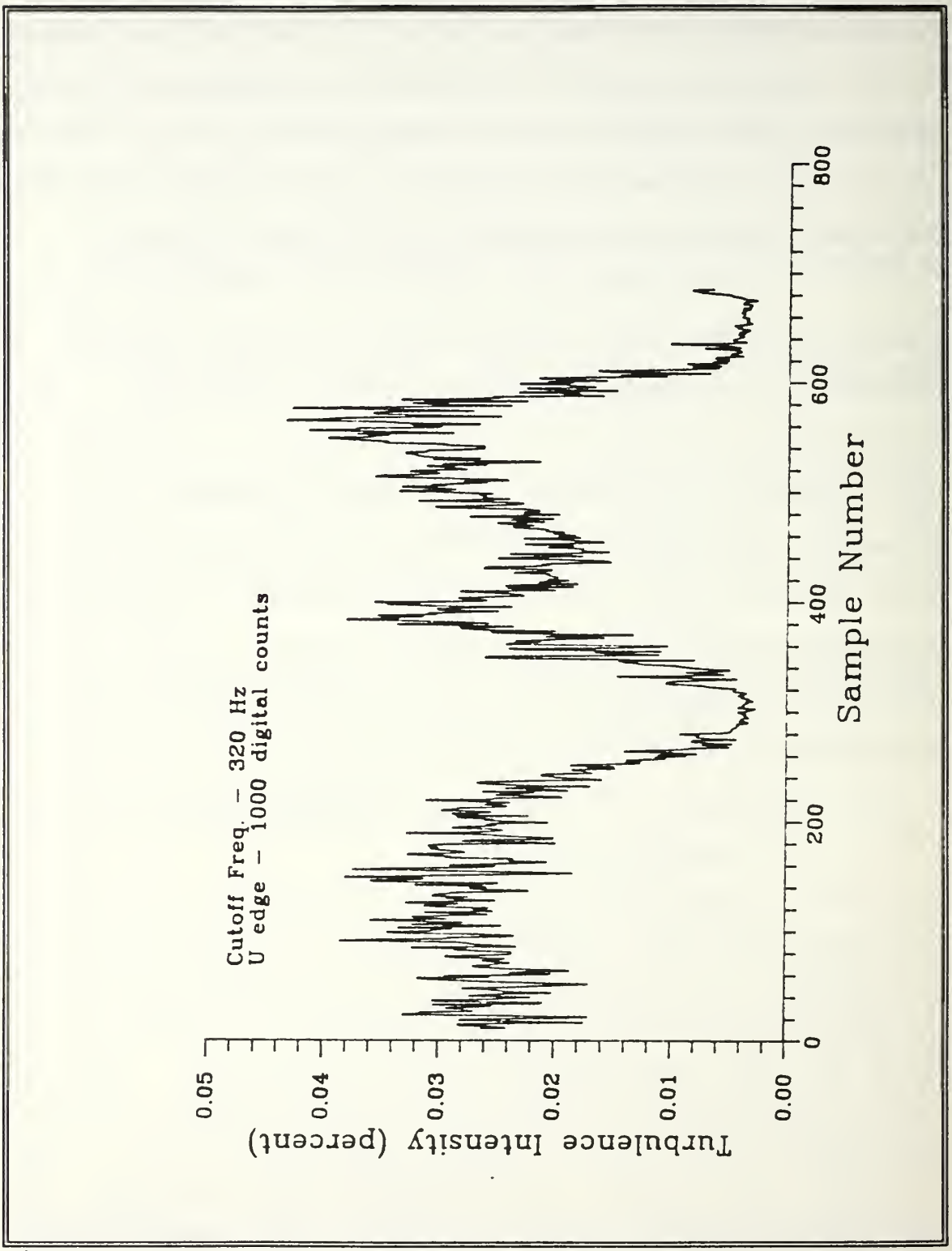

Figure IV-15. RESULT OF PROGRAM TURINTMA ON TEST CASE 1. (TURBULENCE INTENSITY COMPUTED USING 21 POINT MOVING AVERAGE SMOOTHING.)

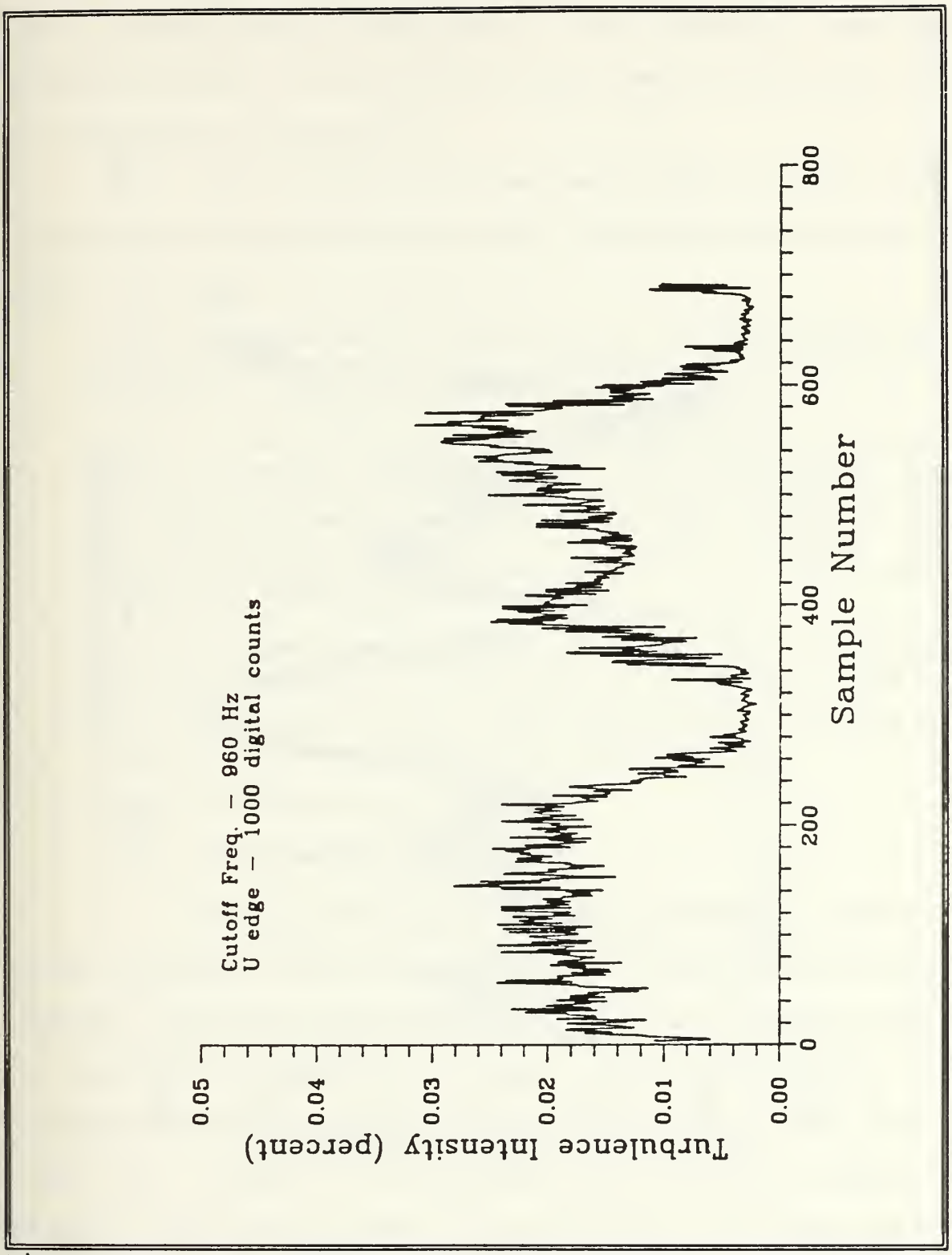

Figure IV-16. RESULT OF PROGRAM TURINTMA ON TEST CASE 1. (TURBULENCE INTENSITY COMPUTED USING 7 POINT MOVING AVERAGE SMOOTHING.)

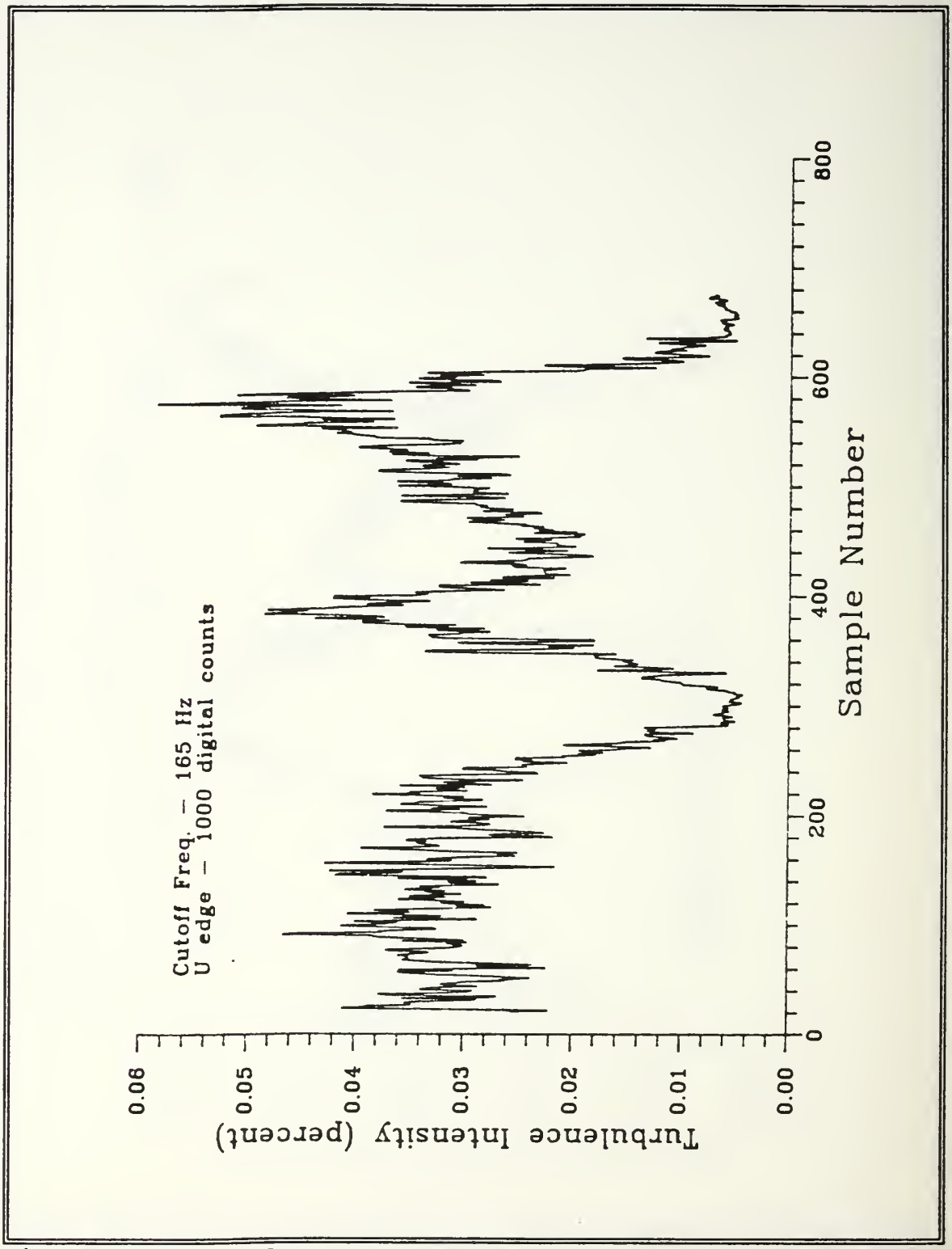

Figure IV-17. RESULT OF PROGRAM TURINTMA ON TEST CASE 1. (TURBULENCE INTENSITY COMPUTED USING 41 POINT MOVING AVERAGE SMOOTHING.)

size and the cutoff frequency. The frequency response characteristics of the low pass filter used in this algorithm are presented in Appendix A.

The algorithm outlined below describes the steps required to determine the turbulence intensity using frequency domain low pass filter smoothing:

- Identify the input and output data file names and enter the desired low pass filter window size corresponding to the desired cutoff frequency of the filter.
- Apply low pass filter smoothing to each column (propeller pulse) of array ensemble. This step requires an FFT routine to accomplish the DFT and IDFT. Subtract the instantaneous velocity from the smoothed velocity to get the fluctuating velocity component. Square the fluctuating velocity component and write the result back into array ENSMBL for storage.
- Perform ensemble averaging on the squared fluctuating velocity to obtain the mean square fluctuating velocity.
- Then compute the square root and normalize (divide by U_{edge}) to obtain the turbulence intensity.
- Write the output file.

b. Implementation and Examples

The algorithm for computing turbulence intensity using frequency domain smoothing was implemented as program TURBIN. The source code and user's guide for program TURBIN is included in Appendix B. The low pass filter and FFT subroutines used in this implementation are from Press, et al. [Ref. 12]. Figure IV-18 presents the results of applying program TURBIN to test case 1 for a 25 point smoothing window (a cutoff frequency of 325 Hz). Figures IV-19 and IV-20

present the results of program TURBIN for an 8 point (f_c = 1015 Hz) and a 51 point (f_c = 160 Hz) smoothing window respectively. These cutoff frequencies were chosen as close as possible to those presented in Figures IV-15 through IV-17. The turbulence intensity results shown in Figures IV-18 through IV-20 correspond to the mean velocity results shown in Figures IV-11 through IV-13 respectively. Again, the cutoff frequency must be chosen with care because the resulting turbulence intensities are strongly dependent upon the cutoff frequency selected.

Since the frequency response characteristics of moving average smoothing are similar to those of frequency domain smoothing (for the particular filter implemented here), the turbulence intensity results are also very similar. Respective comparison of Figures IV-18, IV-19 and IV-20 to Figures IV-15, IV-16 and IV-17 illustrate the similarity. Analogous to the mean velocity algorithms, turbulence intensity from frequency domain smoothing has the advantage of increased flexibility in the choice of the low pass filter characteristics compared to moving average smoothing. Also, frequency domain smoothing has a larger choice of cutoff frequencies, and the beginning and end points are not lost (set to zero) as occurs using the moving average algorithm.

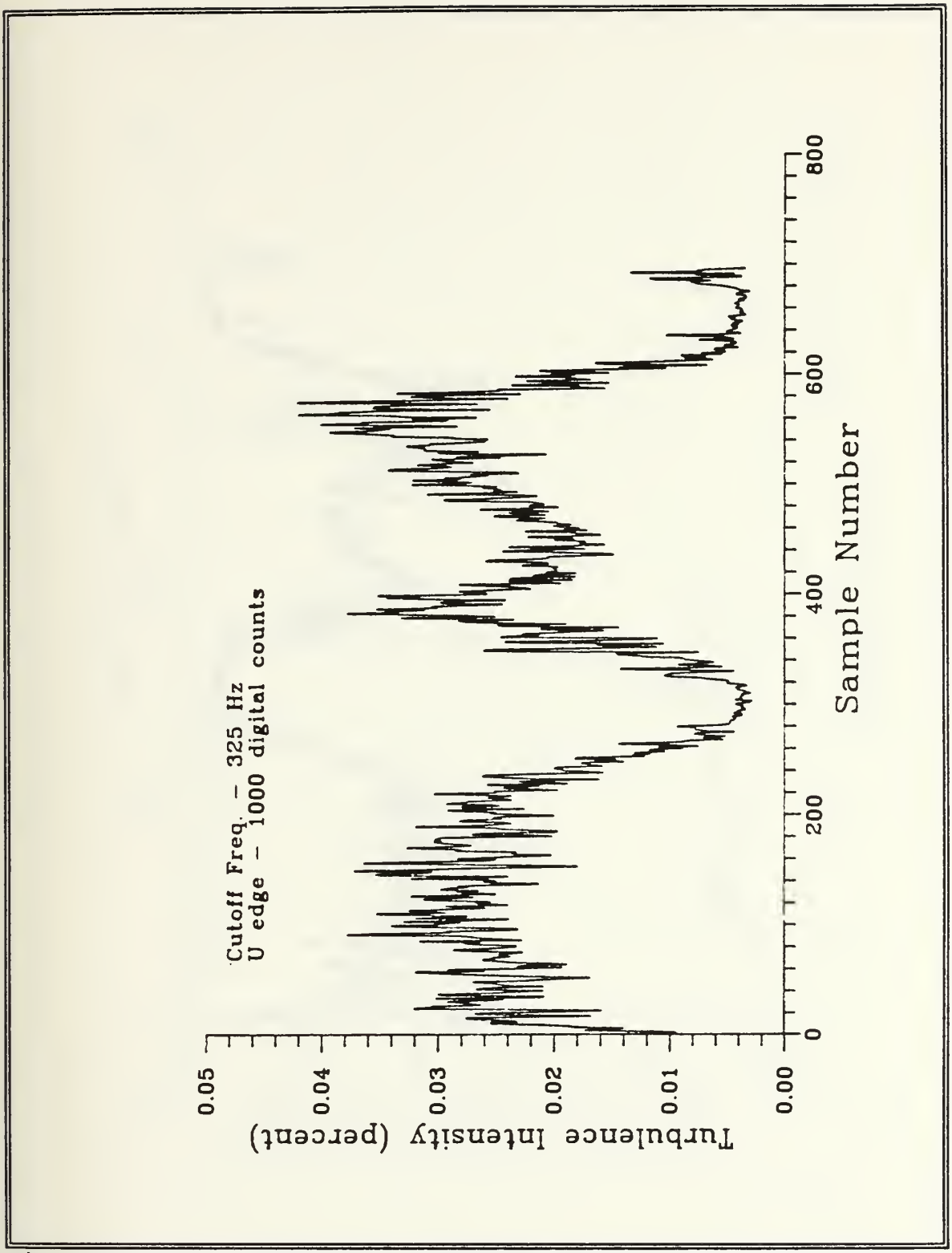

Figure IV-18. RESULT OF PROGRAM TURBIN ON TEST CASE 1. (TURBULENCE INTENSITY COMPUTED USING 25 POINT FREQUENCY DOMAIN LOW PASS FILTERING.)

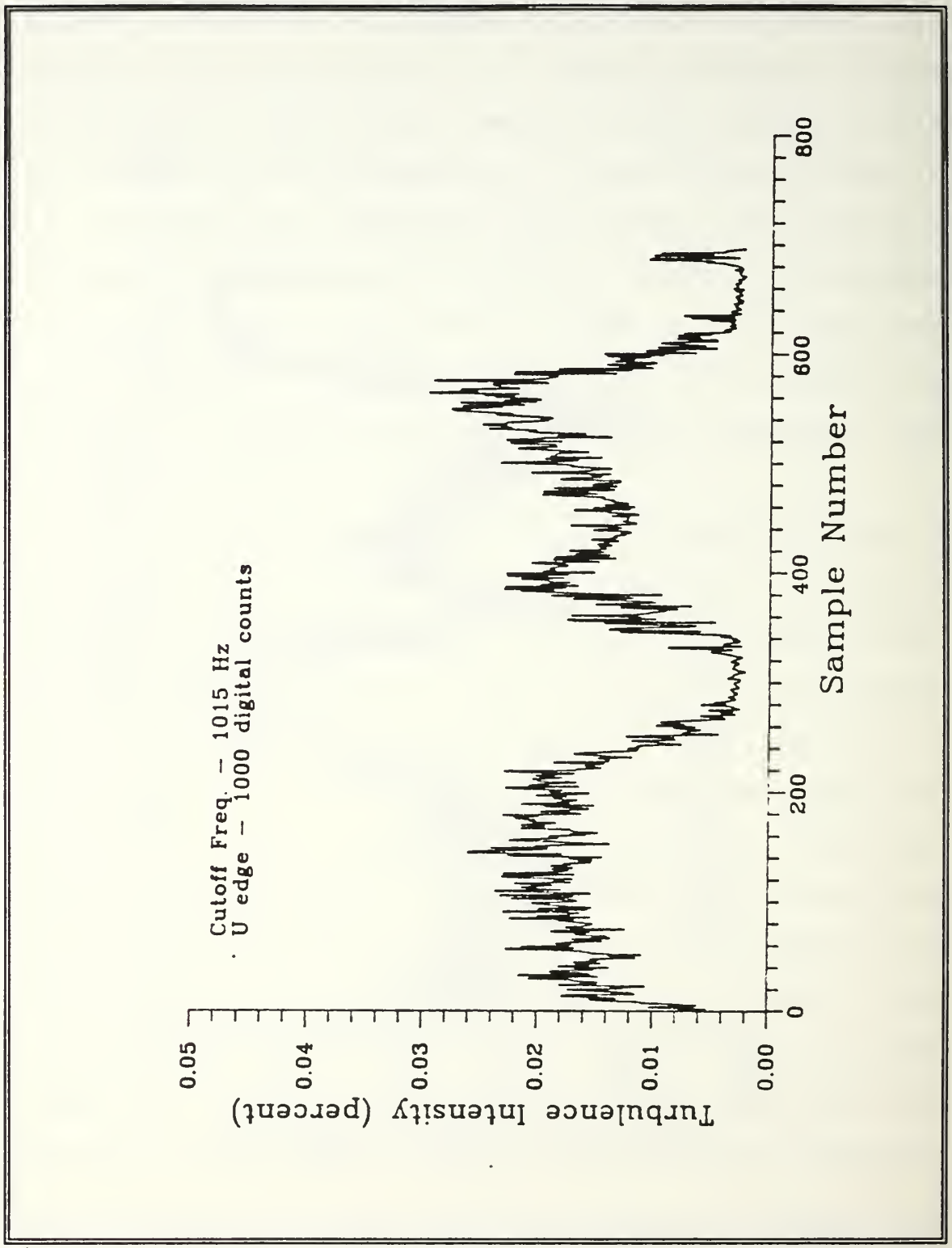

Figure IV-19. RESULT OF PROGRAM TURBIN ON TEST CASE 1. (TURBULENCE INTENSITY COMPUTED USING 8 POINT FREQUENCY DOMAIN LOW PASS FILTERING.)

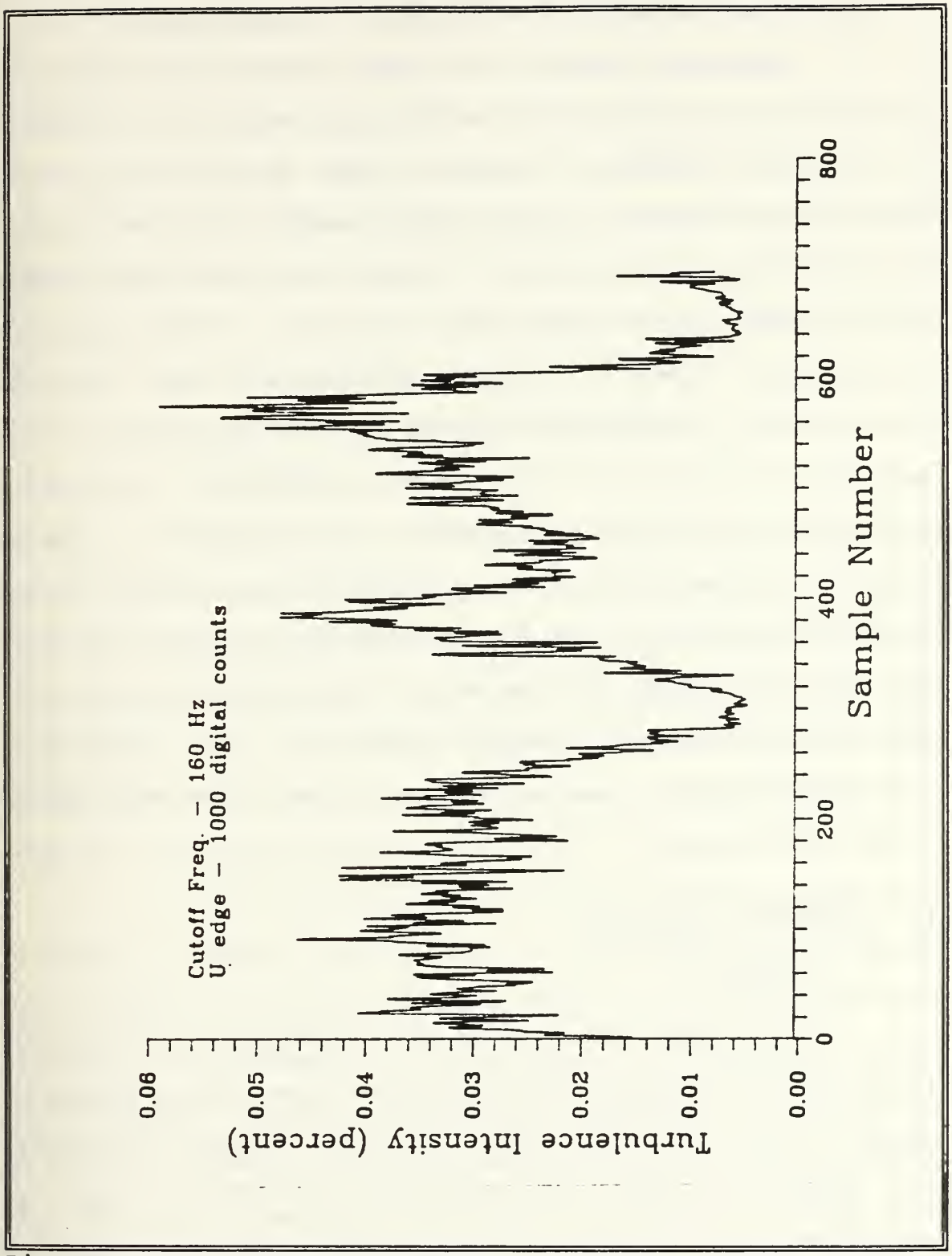

Figure IV-20. RESULT OF PROGRAM TURBIN ON TEST CASE 1. (TURBULENCE INTENSITY COMPUTED USING 51 POINT FREQUENCY DOMAIN LOW PASS FILTERING.)

4. Summary--Turbulence Intensity Determination

Programs TURINTMA and TURBIN provide two alternate methods for computing the non-stationary turbulence intensity of unsteady turbulent boundary layer velocity data. As implemented, the two programs produce nearly identical results for the same cutoff frequency. The advantages of the low pass filter smoothing technique are described in the paragraph immediately above. These advantages lead to the recommendation that program TURBIN be selected as the primary method for determination of the non-stationary turbulence intensity. It is also recommended that alternate frequency domain low pass filters be implemented and tested in the "smoothing" algorithm of program TURBIN to determine if there is any significant difference or improvement in the non-stationary turbulence intensity results. In particular, filters which better approximate the "ideal" low pass filter should be examined.

D. SPECTRAL ANALYSIS

1. Overview

The primary purpose of this section is to describe the technique developed to estimate the time varying frequency spectra of non-stationary data. Spectral analysis algorithms for stationary data are developed initially to gain an understanding of the techniques involved. Two definitions of

the spectral density function were introduced in Chapter III. The first defined the power spectral density (PSD) function as the Fourier transform of the autocorrelation function. The second, more direct definition obtains the PSD by computing the squared magnitude of the Fourier transform of the time history record. The second definition is the one developed in this section. Averaging techniques are employed to obtain a stable, minimum variance spectral estimator.

2. Spectral Analysis of Stationary Data

a. Description of the Algorithm

The spectral estimation scheme developed here is based on the classical "overlapping segments" method developed by Welch and described in Marple [Ref. 10]. The Welch method was selected because it provides an efficient FFT based algorithm. Also, the Welch method does not depend on the assumption of a specific model for the input data (the Welch method is "non-parametric" [Ref. 10].

The idea is to divide the time history record into a number of shorter segments, as illustrated in Figure IV-21. Each segment is multiplied by a window function of the same length as the segment. The purpose of the tapering window function is to minimize the frequency "leakage" effect of the Gibb's phenomenon caused by truncation of the Fourier series (as outlined in section III.B.6.). The "raw" PSD estimate of each windowed segment is then computed per equation III-35.

The power in each segment is corrected for the window function bias by dividing each segment PSD by the window function energy (the sum of the square of each window function term). The PSD estimates for all the segments are then averaged to produce the final Welch PSD estimate. The spectrum is computed from a frequency of zero to one half the sample rate (i.e., the Nyquist frequency, see section III.B.2).

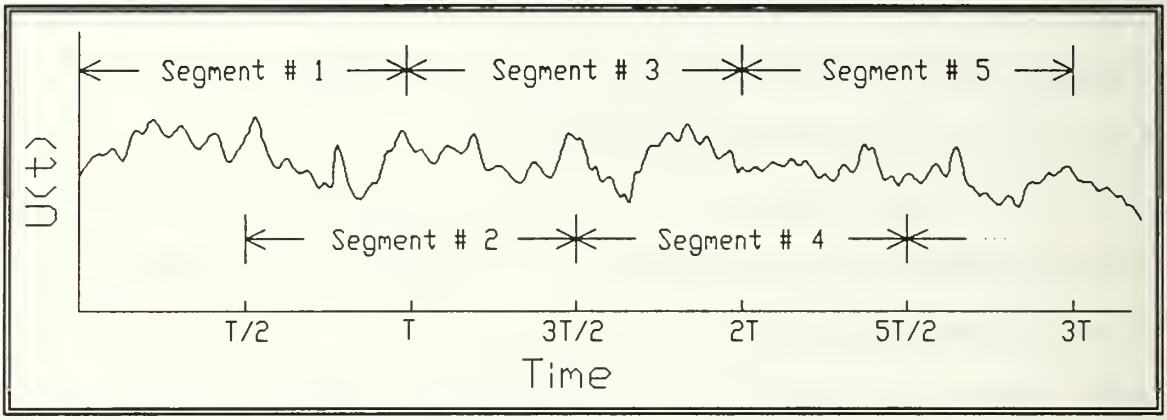

Figure IV-21. SEQUENCE OF OVERLAPPED SEGMENTS OF LENGTH WITH 50% OVERLAPPING.

This "segment and average" method of PSD estimation is necessary because it reduces the variance of the PSD estimate. A raw PSD estimate (equation III-35) has been shown to have a variance on the same order as the estimate itself, making the estimate unreliable at best [Ref. 10]. Marple shows that the variance of the Welch PSD estimator is roughly inversely proportional to the number of segments. So ideally, a large number of segments is desirable. However, the tradeoff is that using a large number of segments reduces the frequency resolution (the number of discrete frequency

bins). The number of positive frequency bins is equal to half the number of data points in the segment. Thus the segments cannot be made too small. For 50% overlapping segments, the relation between the number of positive frequency bins, M, the number of segments, 2K, and the total number of data points in the time series, N, is given by the equation:

$$N = (2K + 1)M \qquad \text{(IV-2)}$$

Since an FFT algorithm is used to compute the DFTs, M must be an integer power of two and N ideally should be an integer power of two. (Note that due to overlapping, some values of K and M yield a value of N which is slightly less than an integer power of 2. The required number of input data points, N, for the spectral analysis algorithm will never be greater than that specified in Table IV-3.) Table IV-3 presents some typical values of the parameters N, M and K. The goal is to choose an adequate number of frequency bins, M with the largest number of segments, K possible. At least 14 segments (K=7) is desirable. The easiest solution to the tradeoff is to collect longer data records. Then both a sufficiently large K and M may be specified.

Many different window functions have been developed to reduce the spectral leakage phenomenon described in Chapter III. Each window function possesses different frequency response characteristics. Window functions are selected based on a tradeoff between frequency resolution loss

Table IV-3. TYPICAL PARAMETER VALUES FOR THE WELCH PSD ESTIMATOR

N	M	K
512	32	7
	64	3
	128	1
1024	32	15
	64	7
	128	3
2048	64	15
	128	7
	256	3
4096	64	31
	128	15
	256	7

(main lobe width) and the amount of leakage suppression (highest side lobe level). An outstanding comparison and discussion of the characteristics and relative merits of most of the popular window functions may be found in the paper by Harris [Ref. 13]. Seven of these window functions were selected to be incorporated into the spectral estimation algorithm and may be selected by the user. Table IV-4 lists the specific window functions with some of their frequency response parameters. The Hamming window or the three term Blackman-Harris window provide a good compromise. The formulas for the window functions may be found in Marple [Ref. 10] or Harris [Ref. 13] and are presented in Appendix B in the FORTRAN function subroutine called FUNCTION WINDOW.

Table IV-4. WINDOW FUNCTIONS

No.	Window	Highest Side Lobe Level (dB)	6 dB Bandwidth (bins)
1	Parzen (triangular)	-27	1.78
2	Rectangular (no window)	-13	1.21
3	Welch (parabolic)	-22	1.55
4	van Hann (cosine^2)	-32	2.00
5	Hamming	-43	1.81
6	3 term Blackman-Harris	-67	1.81
7	4 term Blackman-Harris	-92	2.72

The Welch spectral estimation algorithm is accomplished through the steps listed below:

- Enter the input data file name, the output data file name and the sample rate of the data. Select the window function option and enter the parameters for the number of segments (2K), and the number of frequency bins (M) based on the integer power of two number of data points (N). The input data file must contain at least n data points where N is computed from equation IV-2.

- Allow selection of the option to remove the stationary mean value from the data (i.e., the constant mean velocity of a steady turbulence measurement). Data with a large mean value will yield a spectral estimate with a very large power magnitude at zero frequency. Due to leakage, which windowing reduces but does not eliminate, the large zero frequency power may mask the true spectral characteristics of nearby (low) frequencies. If this option is selected, compute the mean velocity and subtract the mean from each of the instantaneous velocity values. Write the resulting fluctuating velocity component to a scratch file to be used for spectral estimation.

- Compute the spectral estimate using the Welch "overlapping segments" method. This is accomplished using a modified version of a FORTRAN subroutine (SUBROUTINE SPECTRM) extracted from Press et al. [Ref. 12]. The FFT subroutine was also obtained from Reference 12.

- Compute the relative power in dB, of the resulting PSD estimate using the peak value as the reference (zero dB) point. Also compute the percent running total power at each increasing frequency relative to the total power summed over all frequencies.
- Write the output file

b. Implementation and Examples

The spectral estimation algorithm for stationary data described above was implemented as program PSD. The source code and user's guide for program PSD is presented in Appendix B. Figure IV-22 presents the results of program PSD on test case 1. Though test case 1 is not strictly stationary, program PSD will provide a "time averaged" spectral estimate over the length of the time history record. The Hamming spectral window was selected for this example. Program PSD was executed with 256 frequency bins selected (M=256) and 62 segments selected (K=31). These parameter values correspond to 16,128 data points or slightly more than half of the total number of points in the time history record. To obtain a PSD estimate of the entire time history record, the input file may be "zero padded" to a total length equal to an integer power of two data points (32,768 in this case). That is, add zeros on the end of the input file to obtain the proper "power of two" length.

The option to remove the mean was selected for this example (mean value = 317 digital velocity counts) because the resulting zero frequency power spectrum value

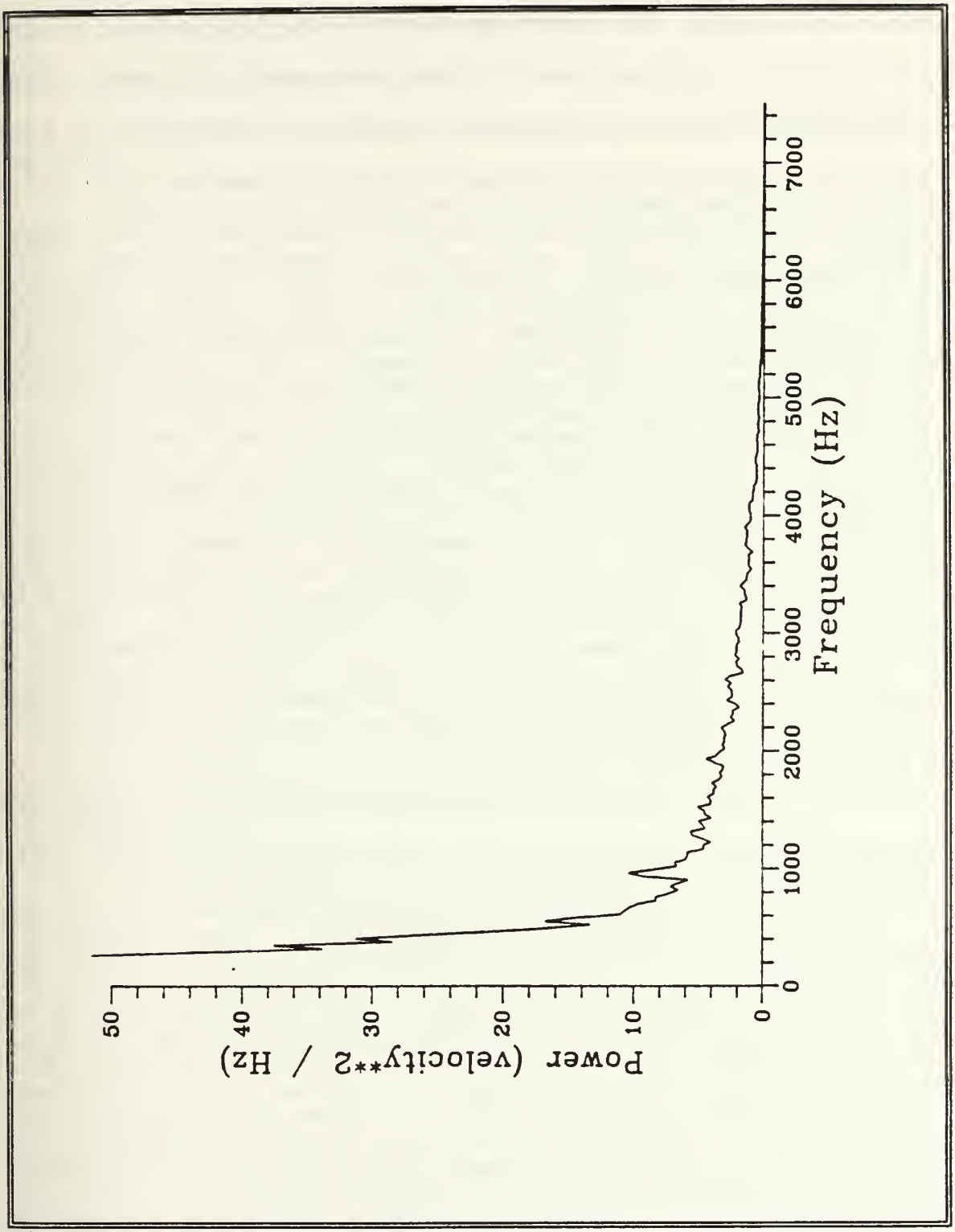

Figure IV-22. SPECTRAL ESTIMATE OF TEST CASE 1 TREATED AS A STATIONARY SIGNAL. (HAMMING WINDOW, STATIONARY MEAN REMOVED, FREQUENCY BINS--256, OVERLAPPED SEGMENTS--31, SAMPLE RATE--15,000 HZ.)

would otherwise be extremely large (on the order of 10) compared to the remainder of the frequency spectrum. Even with the mean removed, the power spectrum contains a high peak centered at the propeller pulse rotational frequency of 40 Hz. The interesting part of the frequency spectrum is well above the propeller pulse frequencies where the turbulent fluctuations lie. Therefore, the spectrum data below 300 Hz. was not shown and the vertical scale was enlarged to better illustrate the high frequency portion of the spectrum. An alternate approach would be to use the low pass filtering algorithm (frequency domain smoothing) to compute and then remove the low frequency velocity component. This would be equivalent to high pass filtering. Finally, program PSD is used without mean removal (since the mean has already been removed) to estimate the spectrum.

Test case 1 was also analyzed using the three term Blackman-Harris window which has lower peak sidelobes than the Hamming window. The resulting spectral estimate plot is not shown because it is virtually identical to Figure IV-22. The three term Blackman-Harris window may be better suited to observing small (low power) spectral peaks which are widely separated from larger peaks. However, Figure IV-22 illustrates the apparent broad band nature of the boundary layer turbulence. Therefore, any of the popular window

functions (options four through seven in Table IV-4) will produce much the same result.

3. Spectral Analysis of Stationary Data--Maximum Entropy Method

An alternate, parametric spectral estimation algorithm was briefly examined. Called the "all poles" or "maximum entropy" method (MEM), this algorithm is based on an autoregressive (AR) parametric model of the time history data [Ref. 10]. The resulting spectral estimates produce spectral peaks which are no longer linearly proportional to the power contained in the underlying sinusoid. Therefore, MEM spectral estimates cannot be averaged as was done using classical methods. The shear volume of data involved with unsteady turbulent boundary layer analysis require the use of averaging. Therefore, MEM spectral estimation cannot be used for the non-stationary application which will be developed in the next section.

Program PSDMEM is presented in Appendix B for the sake of completeness. The program is included "as is" with no detailed explanations or documentation. This program may be used to obtain spectral estimates of stationary data. The maximum entropy method is especially suited to extracting narrow spectral peaks which are close together. (Which is not important to turbulence study.) Also MEM spectral estimation does not require an integer power of two data points and can provide reasonable estimates with a relatively small number

of data points. The program was extracted from Press et al. [Ref. 12]. Additional theory on the maximum entropy method may be found in Marple [Ref. 10].

4. Spectral Analysis of Non-Stationary Data

a. Description of the Algorithm

With non-stationary data, the frequency content of the data varies with time. For the particular application of unsteady periodic boundary layer velocity data, ensemble averaging techniques may be employed on an ensemble of propeller "pulses". The algorithm is somewhat analogous to the moving average scheme. After the data are separated into an ensemble of individual pulses, a raw PSD estimate is computed for a short time segment from the first pulse per equation III-35. This represents the spectrum of the first pulse at the center time point of the segment. Prior to computing the raw spectral estimate, the segment is windowed using an appropriate tapering window just as in the algorithm for stationary data above. Similarly, a raw PSD estimate is computed for the corresponding time segment in every other pulse of the ensemble, windowing the data as before. The collection or raw PSD estimates from the ensemble of short time segments is then averaged and normalized to produce the final "instantaneous" spectral estimate for the ensemble at the time represented by the center of the segment. As before,

the purpose of the normalization is to correct the power for the effect of the tapering window.

The above process can then be repeated for other ensembles of short time segments producing ensemble averaged spectral estimates centered at different times across the ensemble. In this manner, the time varying spectral characteristics may be constructed. The assumption associated with this algorithm (as with the moving average algorithm) is that the data are stationary over the short time segment. This assumption is reasonable since the mean varies slowly compared to the high frequency fluctuating velocity component. However, the time segment must be long enough to achieve the desired resolution in the frequency spectrum. If the frequency content of the data does change in time over the segment, then the spectral estimate is still useful. The spectral estimate then represents the average frequency content of the data over the time segment rather than the instantaneous spectrum at the center of the time segment.

Evaluating the tradeoff between short segment length and desired frequency resolution led to the choice of 256 point segments. This provides 128 positive frequencies in the frequency spectrum which is adequate. Also, 256 points represents roughly 1/3 to 1/6 the length of the entire pulse for sample rates of 15,000 and 30,000 Hz. respectively. A segment which is 1/3 the length of the entire pulse is

considered marginal for the "stationary short segment" assumption. Increasing the sample rate to 20,000 Hz. or more is an easy solution to the problem. Another equivalent alternative is to decrease the frequency resolution to 64 positive frequencies which corresponds to a segment width of 128 points. Yet another option is to maintain a 256 point wide window, but (similar to the moving average window) increase the resolution in time by overlapping the ensemble of segments such that the final spectral estimates are centered on time points which are less than 256 points apart (say, 50 or 100 points for example). These options are not mutually exclusive and may be employed together.

The variance of the final spectral estimate of a segment of the ensemble is inversely proportional to the number of pulses which comprise the ensemble. This is analogous to the stationary case except the segments are in the ensemble sense and not overlapping segments in time. Therefore 30 to 50 pulses should be adequate to obtain a reasonable stable spectral estimate. This would roughly correspond to a K parameter in the range of 15 to 25.

This non-stationary spectral estimation algorithm provides the same seven tapering window options which were described above for the stationary case. The window functions and some of their frequency response characteristics are presented inn Table IV-4. As before, the Hamming window or

the three term Blackman-Harris window are recommended for the broad band turbulence data.

The non-stationary spectral estimation algorithm is summarized using the following steps listed below:

- Enter the input data file name, the output data file name and the sample rate of the data.
- Separate the input time history record into an ensemble of individual pulses. As previously described, each column of array ENSMBL represents a single propeller pulse.
- Allow selection of the option to remove the mean value from the data (i.e., remove the mean velocity of each individual segment). Data with a large mean value will yield a spectral estimate with a very large zero frequency value. This may mask the true spectral estimate of the nearby (low) frequencies.
- Allow selection of the tapering window option.
- Allow selection of one of three options for segmenting the ensemble. The first option segments the ensemble based on an input value for the number of data points between the beginning of each segment (i.e., the amount of time between each PSD). The second option segments the data based on an input value for the number of segments across the ensemble of pulses (i.e., the number of PSDs across the ensemble). The third option creates a single segment ensemble based on an input time (sample) value of the center of the segment (i.e., choose the location for a single PSD).
- Remove the mean of each segment if selected above. Then, compute the raw spectral estimate of the corresponding segment of each pulse, windowing the data prior to calling the FFT subroutine. The FFT subroutine was extracted from Press, et al. [Ref. 12]. Now average the raw PSDs from each pulse and normalize to correct for the window.
- Compute the relative power of the final segment PSD in dB using the peak value as the reference (zero dB) point. Also compute the percent running total power at each

increasing frequency relative to the total power summed over all frequencies.

- Write the output file.

b. Implementation and Examples

The algorithm for non-stationary spectral estimation was implemented as program ENSPSDAV. The source code and user's guide is presented in Appendix B. Figure IV-23 presents the results of applying program ENSPSDAV to test case 1. Figure IV-23 presents frequency vs. power magnitude for three different time slice ensemble segments. All three segments were computed using the Hamming window and with the segment mean removed. The first ensemble segment, indicated by the solid line, is centered at sample 140, which is roughly the center of a portion of turbulent flow for the 41 pulses which comprise test case 1 (see Figure IV-1 for the relative position of the segments). The second ensemble segment, indicated by the short dashed line, is centered at sample 300, which is roughly the center of a portion of laminar flow. Observe the significantly lower power levels exhibited by the laminar segment at all frequencies above approximately 200 Hz. The third ensemble segment, indicated by the long dashed line, is centered at sample 500. Observe that, as expected, the power level has returned to that associated with turbulent flow. Figure IV-23 graphically illustrates the time dependent nature of the frequency spectra of non-stationary data.

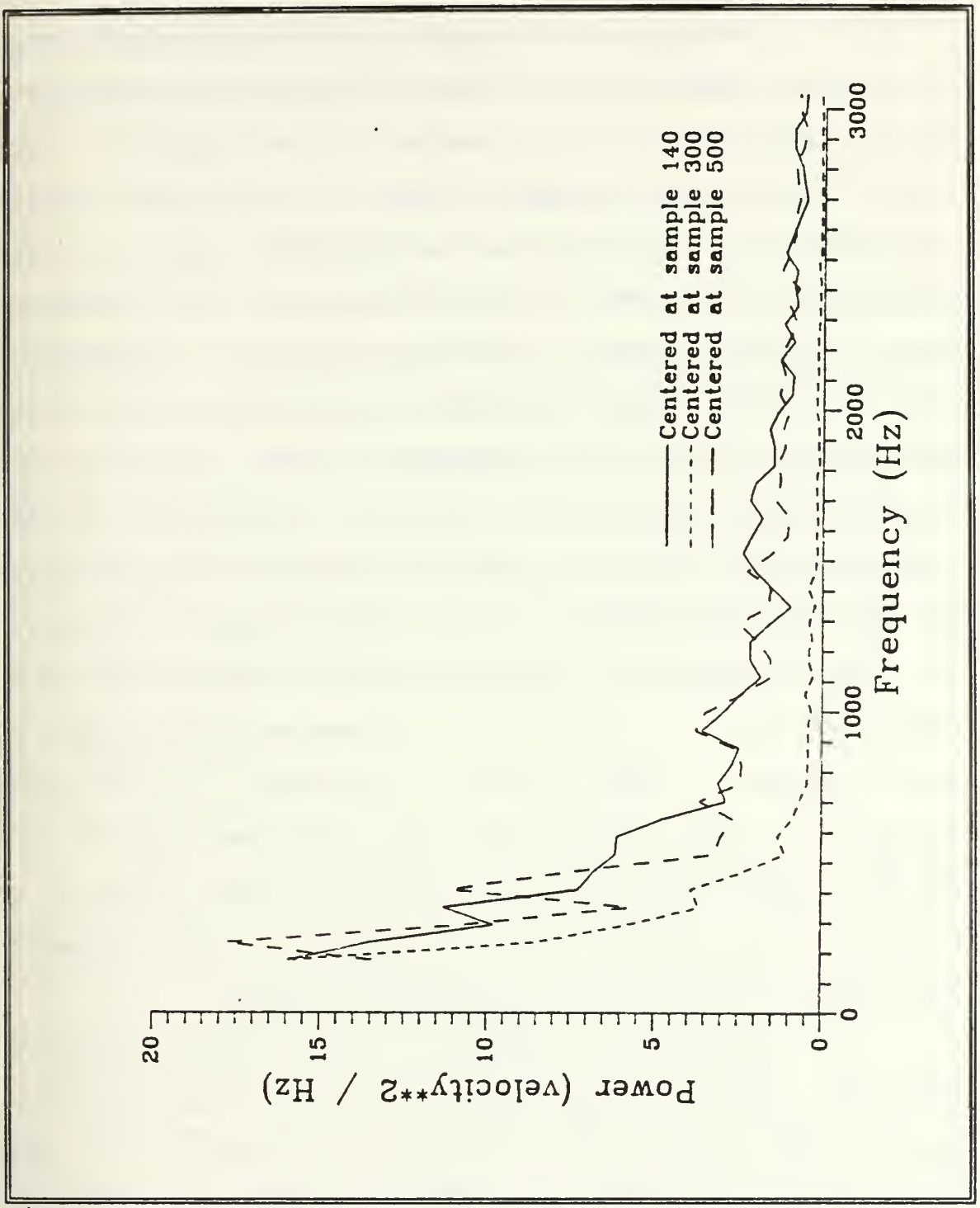

Figure IV-23. SPECTRAL ESTIMATE OF THREE DIFFERENT ENSEMBLE SEGMENTS FROM TEST CASE 1. (HAMMING WINDOW, MEAN REMOVED.) SAMPLE RATE = 15,000 HZ.

The option to remove the mean was selected for this example, as with the stationary example discussed above, because the resulting zero frequency power spectrum value would otherwise be extremely large (on the order of 10) compared to the remainder of the frequency spectrum. The interesting part of the frequency spectrum is well above the propeller pulse frequencies where the turbulent fluctuations lie. Therefore the spectrum data below 200 Hz was not shown and the vertical scale was enlarged to better illustrate the high frequency portion of the spectrum. As discussed for the stationary case above, an alternate approach would be to use the low pass filtering algorithm (frequency domain smoothing) to compute and then remove the low frequency velocity component from each pulse in the ensemble (each column in array ENSMBL). This would be equivalent to high pass filtering. Then proceed with the algorithm without mean removal (since the "mean" has already been removed) to estimate the spectrum of the remaining high frequency fluctuating components of the ensemble segment.

Examination of Figure IV-1, IV-2 and IV-3 suggests that the spectral characteristics of the unsteady boundary layer transition abruptly from laminar to turbulent flow and back again as opposed to a gradual transition. If the ensemble segment crosses the transition point, the resulting spectrum is an average of the frequency content of the two

regions on either side of the transition point. A series of 256 point wide ensemble segments moving across a transition point will produce a series of spectra which shows a gradual change from low power (laminar flow) to high power (turbulent flow). This phenomenon is due to the assumption of stationarity over a short time segment. If the segment encompasses a transition point, then the short segment stationarity assumption is violated (as discussed in paragraph D.4.a. above). The user should exercise care to insure that the ensemble segment does not cross a transition point. As a minumum, positioning the transition point near one edge of the ensemble segment will allow the tapering window function to minimize the effect of the abrupt transition.

E. CORRELATION

1. Overview

The autocorrelation and cross correlation functions of stationary data were explicitly defined in Chapter III by equations III-17, III-20 and III-21. Equation III-32 presents an alternate definition of correlation (auto or cross) which is obtained indirectly through the use of the DFT. Algorithms are developed and implemented in this section for both the explicit and indirect methods. Then algorithms are discussed for the non-stationary case. Time limitations precluded implementation of the algorithms for non-stationary data.

2. Correlation of Stationary Data--Explicit Method

a. Description of Algorithm

The algorithm for computing either autocorrelation or cross correlation is essentially a direct implementation of equations III-17, III-20 and III-21. The algorithm is accomplished through the following steps:

- Identify the input and output data file names and specify the option to compute autocorrelation or cross correlation.
- If autocorrelation is selected, the input file must consist of a sequential file containing the sampled values of the random variable to be analyzed (e.g., instantaneous velocity, u or v). If cross correlation is selected, the input file must consist of a sequential file containing alternating values of the two random variables to be analyzed (e.g., instantaneous velocities u and v). The first variable of each pair of samples is the variable which is lagged. (For example enter alternating samples of u and v beginning with u to compute R_{vu}).
- Select the number of lags (the maximum γ) for which to compute the correlation. (Enter as a fraction of the length of the input data file).
- Remove the mean of the stationary data to obtain the fluctuating component.
- Compute the unbiased correlation function (auto or cross depending on the option selected). This step uses subroutine CORMAR which was extracted from Marple [Ref. 10].
- Write the output file.

b. Implementation and Examples

Figure IV-24 presents test case 2 which is an example of steady (stationary) turbulence. This particular example was generated by measuring the velocity of the air

flow behind a grid or screen. The mean velocity was 95 feet/second and the sample rate was 1000 Hz.

Test case 2 was used as the example input to program CORREX which is the implementation of the explicit method for autocorrelation and cross correlation. The source code and user's guide for program CORREX are presented in Appendix B. Program CORREX was implemented to compute the correlation functions for positive lag values only. The autocorrelation function of a real time series is symmetric about the origin. Figure IV-25 presents the results of computing the autocorrelation function of test case 2 using program CORREX. As expected with random data, the autocorrelation of the fluctuating component at zero lag is a large value (the mean square of the fluctuating component). The autocorrelation function falls rapidly to oscillate about zero. In turbulence analysis, the autocorrelation function is integrated over all lags to obtain a measure of the "turbulence length scale" outlined in Chapter II. This essentially becomes the area under the initial peak.

There is no actual two-component velocity data available with which to demonstrate the cross correlation option of program CORREX. Therefore two-component data were simulated by using the single velocity component of test case 2 twice, with the second velocity "component" (v) a time shifted version of the u component. The "v component"

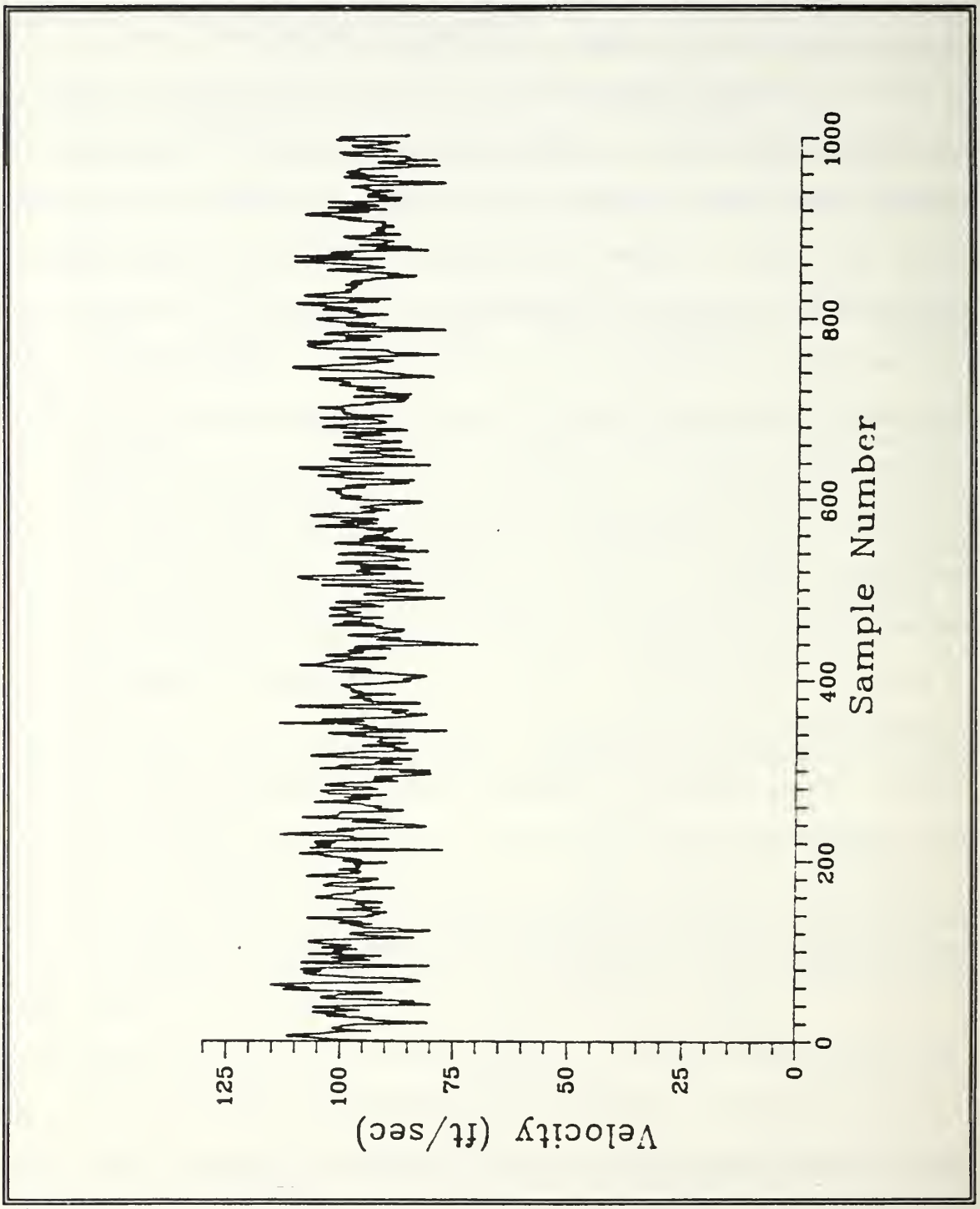

Figure IV-24. TEST CASE 2 (AN EXAMPLE OF STATIONARY TURBULENCE. SAMPLE RATE--1000 HZ, MEAN VELOCITY--95.4 FT/SEC.)

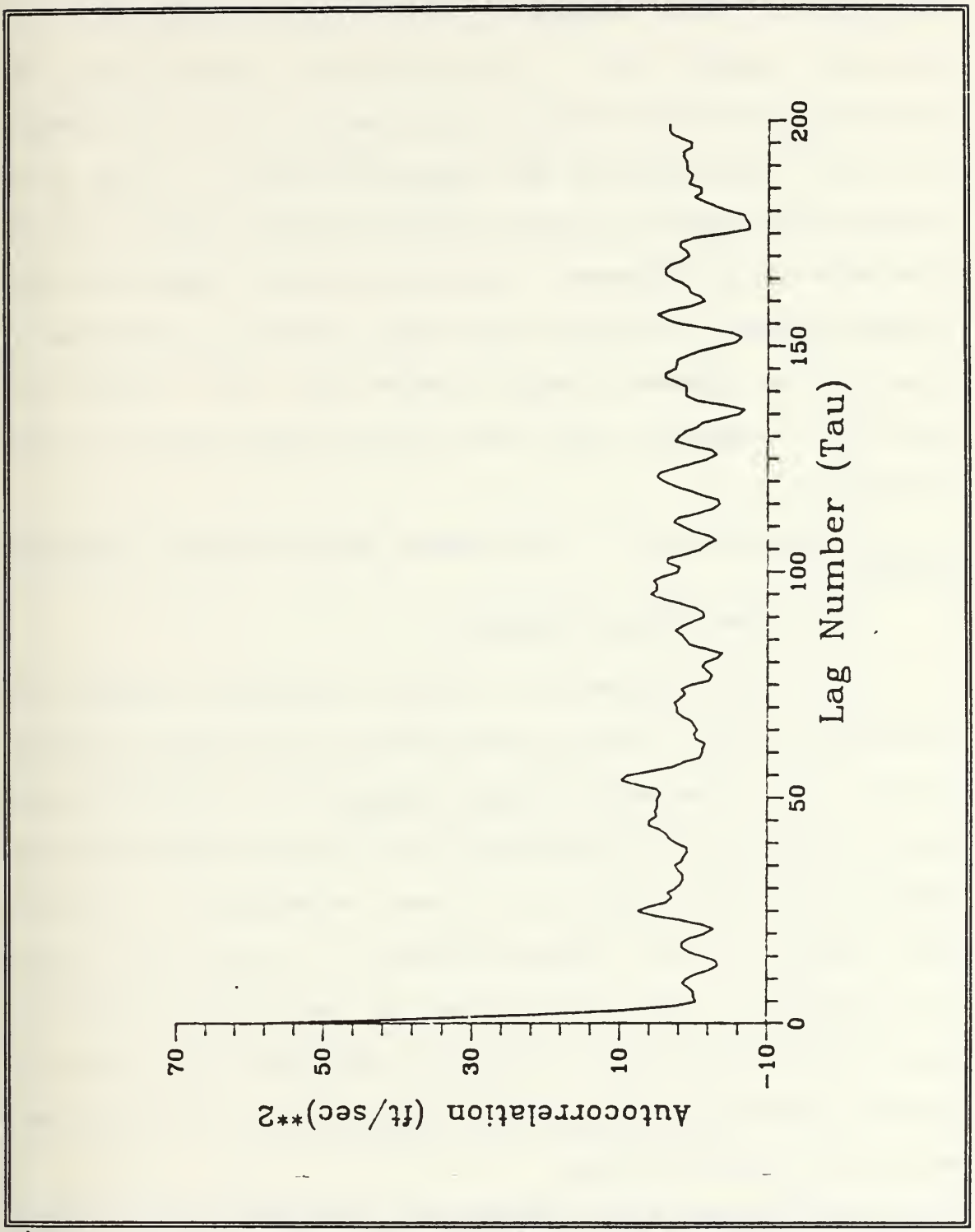

Figure IV-25. RESULT OF PROGRAM CORREX ON TEST CASE 2 (AUTOCORRELATION OF STATIONARY TURBULENCE).

consists of the u component shifted by 50 samples. The expected result is a time shifted version of the autocorrelation with the peak occurring at a lag of 50 instead of zero. Illustrating the expected result, Figure IV-26 presents the results of the cross correlation function of the simulated two component velocity signals computed using program CORREX. To obtain the cross correlation function at negative lag numbers, simply reverse the u and v input data vectors. In general, the cross correlation function is not symmetric.

3. Correlation of Stationary Data--Fourier Transform Method

a. Description of Algorithm

The indirect or Fourier transform method for computing auto and cross correlation was outlined in section III.B.4. The idea is to compute the circular correlation per equation III-31, then normalize the circular correlation per equation III-32 to obtain the linear correlation. The input time series (u for autocorrelation, u and v for cross correlation) is zero padded prior to computing the circular correlation so that the correlation function at positive and negative lags may be separated. The algorithm is accomplished using the following steps:

- Identify the input and output data file names and specify the option to compute autocorrelation or cross correlation.

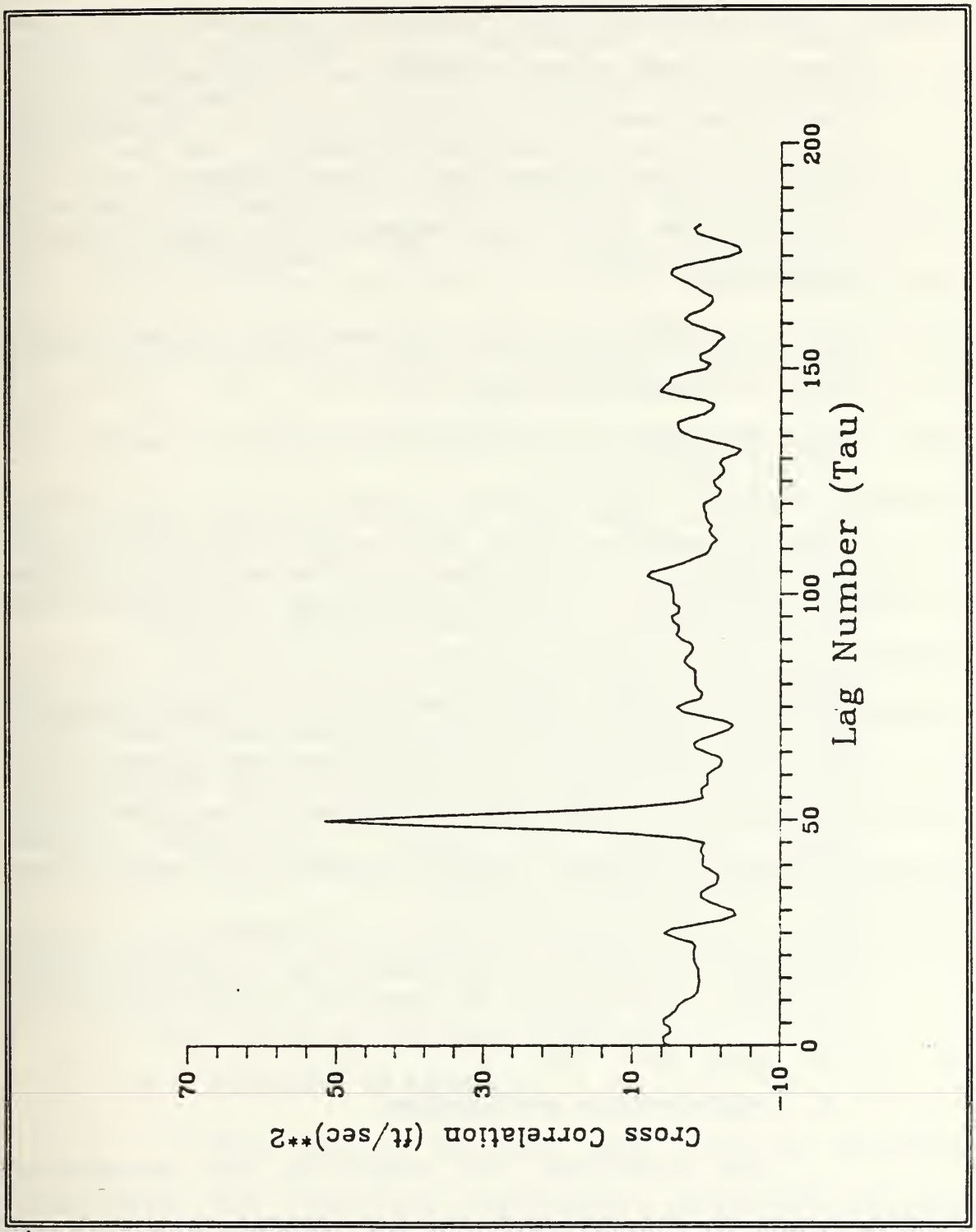

Figure IV-26. RESULT OF PROGRAM CORREX ON TEST CASE 2 (SIMULATED CROSS CORRELATION OF STATIONARY TURBULENCE).

- If autocorrelation is selected, the input file must consist of a sequential file containing the sampled values of the random variable to be analyzed (e.g., instantaneous velocity, u or v). If cross correlation is selected, the input file must consist of a sequential file containing alternating values of the two random variables to be analyzed (e.g., instantaneous velocities u and v). The first variable of each pair of samples is the variable which is lagged. (For example enter alternating samples of u and v beginning with u to compute R_{vu}).

- Select the number of lags (the maximum τ) for which to compute the correlation. (Enter as a fraction of the length of the input data file).

- Remove the mean of the stationary data to obtain the fluctuating component.

- Zero pad the data with at least as many zeros as the number of lags for which the correlation function will be computed (i.e., the number of zeros equals the maximum number of lags). If cross correlation was selected, then zero pad both input time series (e.g., u and v).

- Compute the circular correlation function (auto or cross depending on the option selected). This step uses subroutine CORREL which was extracted from Press, et al. [Ref. 12]. The circular correlation subroutine calls FFT routines which were also extracted from Reference 12. Note that this subroutine computes N times the result given by equation III-31 (where N is the number of points in the time series). Also note that both positive and negative lags are evaluated.

- Normalize the circular correlation by dividing the result of the previous step by the factor (N-r) where r is the lag number. This yields the desired linear correlation.

- Write the output file.

b. Implementation and Examples

The algorithm for computing the correlation functions using the indirect Fourier transform method was implemented as program CORFFT. The source code and user's guide are presented in Appendix B. Test case 2 was used as the example input to program CORFFT. The results are

virtually identical to those obtained from program CORREX, (see Figures IV-25 and IV-26) so no additional figures are presented.

The primary advantage to program CORFFT is increased computational efficiency over the explicit method. The explicit method requires on the order of N^2 floating point operations to compute the correlation function. Using a FFT algorithm, the Fourier transform method computes the correlation function in roughly N log(N) floating point operations. As an example, using the explicit method (program CORREX), the autocorrelation example of test case 2 required 26.7 seconds of computation time. A 50% time savings was achieved using the Fourier transform method)program CORFFT), which required only 13.4 seconds to obtain the same result. Even larger percentage time savings may be expected with longer input data files. (Test case 2 is only 1000 data points long.) Therefore program CORFFT is the recommended choice for computing correlation functions of stationary turbulence data.

4. Correlation of Non-Stationary Data

a. Description of Algorithm

This section outlines some ideas on potential algorithms for computing correlation functions of non-stationary data. Time limitations prevented completion of the effort to develop these ideas into working computer programs.

Recall that the correlation function and the spectral density function form Fourier transform pairs. Therefore the algorithm for estimating non-stationary correlation functions should parallel that developed for non-stationary spectral estimation. One approach is proposed for computing autocorrelation. Simply compute the inverse DFT of the segment spectrum computed using program ENPSDAVG presented in section IV.C.4.

However, a general algorithm for either autocorrelation or cross correlation is desired. Without repeating section IV.C.4., the basic idea is to divide the ensemble of non-stationary data into short time segments where the assumption of local stationarity may be applied. Positive lags up to half the length of the segment may be computed. The correlation function of the corresponding segment from each pulse is computed, then ensemble averaged to obtain the final correlation function for the ensemble of segments.

As with the non-stationary spectral analysis, an important issue to consider in non-stationary correlation is the length of the segment. If the segment is too short, then a sufficient number of lags cannot be computed in order to characterize the turbulence length scale (autocorrelation). If the segments are too long, then the assumption of local stationarity is violated. At this point it is not clear that 128 positive lags (corresponding to a 256 point wide segment) are sufficient to derive an accurate turbulence length scale.

or increasing the number of pulses so that a pure ensemble averaging algorithm may be applied. Again, the tradeoff is computer memory and storage limitations.

Computer memory limitations become critical when cross correlation functions are desired. The array containing the ensemble of non-stationary data must now be twice as large to contain two input data vectors (u and v) instead of one. Rather than storing all of the data in computer memory at once, techniques may need to be developed to handle the data sequentially or in smaller blocks.

The basic algorithm for computing non-stationary correlation functions is outlined in the following steps:

- Enter the input data file name, the output data file name and the sample rate of the data.
- Separate the input time history record into an ensemble of individual pulses. As previously described, each column of array ENSMBL represents a single propeller pulse.
- Allow selection of one of three options for segmenting the ensemble. The first option segments the ensemble based on an input value for the number of data points between the beginning of each segment (i.e., the amount of time between each PSD). The second option segments the data based on an input value for the number of segments across the ensemble of pulses (i.e., the number of PSDs across the ensemble). The third option creates a single segment ensemble based on an input time (sample) value of the center of the segment (i.e., choose the location for a single PSD).
- Remove the mean of each segment. Then zero pad each segment such that the positive and negative lags may be separated.
- Compute the circular correlation estimate of the corresponding segment of each pulse. Average the correlation function from each pulse segment and

normalize to obtain linear correlation from circular correlation.

- Write the output file.

V. CONCLUSIONS AND RECOMMENDATIONS

A data analysis system was developed to support a planned wind tunnel investigation into the nature of unsteady turbulent boundary layers. A series of algorithms were developed and implemented to compute the statistical parameters which the aerodynamicist uses to characterize the nature of turbulent fluid flow. Specifically, the aerodynamicist will use these data analysis systems to analyze fluid velocity measurements obtained in the boundary layer of an airfoil subjected to periodic turbulent wake disturbances such as in a propeller slipstream. The statistical parameters used to characterize the unsteady turbulent boundary layer include the mean velocity, turbulence intensity, power spectral density function and the autocorrelation or cross correlation functions. Unsteady turbulent fluid flow is a non-stationary random process. Therefore the statistical parameters of interest are time varying functions and require special non-stationary techniques to estimate.

The following paragraphs summarize the specific algorithms included in the data analysis system. Where alternate algorithms are proposed for a single statistical parameter, the "best" method is recommended base on specified criteria.

A. MEAN VELOCITY

Four different algorithms were developed and implemented to determine the non-stationary mean velocity at a point in the boundary layer. Program ENSEMBL estimates the non-stationary mean by performing ensemble averaging on an ensemble of propeller blade wake passages. Storage and computer memory limitations preclude collecting enough data to obtain an adequate estimate of the mean using only ensemble averaging. Program MOVAVE performs a moving time average on each pulse in the ensemble to obtain a "local mean" prior to ensemble averaging. Program SMUMEAN performs frequency domain low pass filtering on each pulse to obtain a "local mean" prior to ensemble averaging. Program MOVAVE and SMUMEAN both yield improved estimates of the non-stationary mean. Program SMUMEAN achieves the following advantages:

- Increased flexibility due to a large number of available cutoff frequencies.
- Ability to implement alternate low pass filters, if desired.
- No loss of data at the beginning and end of the ensemble average data record.

Program MEANSMU performs the ensemble averaging operation prior to frequency domain low pass filtering. This achieves the identical result as program SMUMEAN with an approximately 75% reduction in computation time. Therefore, program MEANSMU is the recommended method for determining the non-stationary

mean velocity. Program MEANSMU (and SMUMEAN) was implemented with only one type of low pass filter. Therefore, it is further recommended that alternate frequency domain low pass filters be implemented and tested in the smoothing algorithm of program MEANSMU to determine if there is any significant difference or improvement in the non-stationary mean velocity estimate. In particular, filters which better approximate the ideal low pass filter should be examined.

B. TURBULENCE INTENSITY

Turbulence intensity is a measure of the amount of variation of the fluctuating velocity component about the local mean velocity. Two different algorithms were implemented based on two methods for estimating the local mean. Program TURINTMA computes the local mean via a moving average. Program TURBIN computes the local mean via frequency domain low pass filtering. Programs TURINTMA and TURBIN produce nearly identical turbulence intensity results because the frequency response characteristics of the two different smoothing algorithms are similar. The smoothing algorithms are the same as those used in the mean velocity algorithms, so the same advantages described above apply to program TURBIN. Therefore, program TURBIN is the recommended method for estimating non-stationary turbulence intensity. It is further recommended that alternate frequency domain low pass

filters be implemented and tested in the smoothing algorithm of program TURBIN to determine if there is any significant difference or improvement in the non-stationary turbulence intensity estimates. In particular, filters which better approximate the ideal low pass filter should be examined.

C. SPECTRAL ANALYSIS

Two programs were developed to estimate the spectral characteristics of the turbulent boundary layer velocity data. Program PSD was developed for steady (stationary) turbulence. The stationary case was examined first to gain an understanding of the techniques involved. Program PSD is an implementation of the classical Welch "overlapping segments" periodogram method of spectral estimation. The Welch method is non-parametric and as such, does not depend on the assumption of a model for the random data in order to obtain accurate spectral estimates. Several spectral window function options are included for leakage reduction. Window option 5 (Hamming window) or 6 (3 term Blackman-Harris window) are the recommended choices of window functions for the generally broad band turbulence data. These window functions provide a good compromise between the ideals of narrow main lobe width and low side lobe level. However, due to the broad band nature of the data, any of the popular windows (options 4 through 7 in Table IV-4) will produce similar results.

Program PSD also provides an option for removal of the stationary mean prior to estimating the spectrum. This option should be selected for data with a large mean value. Due to spectral leakage (which windowing reduces but does not eliminate) the large zero frequency power may mask the true power of nearby (low) frequencies. It is further recommended that an alternate option for mean removal be implemented. Rather than removing only the mean value, an option should be implemented which will high pass filter the data and remove all low frequency content below the frequencies of interest.

Program ENSPSDAV was developed to estimate the time varying non-stationary spectral characteristics of an ensemble of propeller blade wake passages. This program computes a spectral estimate for a short time segment of each propeller pulse in the ensemble. The data are reasonably assumed to be stationary over a short segment. The program then ensemble averages the individual spectral estimates to obtain the final estimate for the short time segment of the ensemble. The non-stationary (time varying) spectral characteristics are constructed by examining different short time segments.

The same options for window functions and mean removal are provided in program ENSPSDAV as are provided in program PSD discussed above. The same recommendations for window function choice and an alternate mean removal method also apply to program ENSPSDAV. It is further recommended that the user of

program ENSPSDAV exercise care in the selection of the short time segment locations. The user should insure that the short time segment does not cross a transition point between laminar and turbulent flow, thus violating the assumption of stationarity over the short segment. If a short segment cannot be located without including a transition point, then the transition point should be located near the beginning or end of the segment such that the window function will minimize the effect of the abrupt transition point.

D. CORRELATION

Two programs were developed to estimate the autocorrelation or cross correlation of stationary data. Program CORREX is a direct implementation based on the explicit definition of autocorrelation and cross correlation. Program CORFFT estimates the autocorrelation or cross correlation using the indirect Fourier transform method. These two programs produce virtually identical results. Program CORFFT has the primary advantage of increased computational efficiency. For a test case data record which is 1024 points in length, a 50% computational time savings was achieved. Greater percentage savings are expected for longer data records. Therefore, program CORFFT is the recommended choice for estimating correlation functions of stationary turbulence data.

Some ideas were presented for potential algorithms to estimate the correlation functions of non-stationary (unsteady) turbulence data. Time limitations prevented the development of these ideas into working computer programs. It is recommended that the algorithms for estimating the non-stationary autocorrelation or cross correlation functions presented in Chapter IV be developed into working computer programs.

APPENDIX A

FREQUENCY RESPONSE CHARACTERISTICS OF THE SMOOTHING ALGORITHMS

A. FREQUENCY RESPONSE OF THE MOVING AVERAGE SMOOTHING FILTER

A moving average smoothing filter is one form of a non-recursive time domain low pass digital filter. Moving average smoothing involves computing a time average over a short segment of a time series. The average of the short segment represents the smoothed value for the center point of the segment. This is equivalent to the midpoint of the linear least squares curve fit through the segment of points. For a discrete time series $u(t_i)$ with equally spaced samples, the moving average filter is given by the equation:

$$\bar{u}_{i_{LOCAL}} = \frac{1}{2N+1} \sum_{m=-N}^{N} u_{i-m} \qquad \text{(A-1)}$$

where $u_{i_{LOCAL}}$ is the smoothed value or "local mean" corresponding to the instantaneous value $u(t_i)$. L is the length of the moving average window (an odd number) such that $L = 2N + 1$. In other words, N is the number of points on each side of the i'th "pivot point". The index m runs from -N to N. [Ref. 9]

To examine this digital filter in the frequency domain, examine the transfer function of the digital filter. The transfer function of the digital filter is the eigenvalue of equation A-1. To find the transfer function (eigenvalue), substitute an eigenfunction for the input time series $u(t_i)$, then solve for the eigenvalue. The trigonometric sinusoids are the eigenfunctions of an equally spaced sampled process and form an orthogonal basis spanning the finite time series interval [Ref. 9]. Writing the trigonometric sinusoids in complex exponential form, let

$$u(t_i) = e^{j\omega t_i} \qquad \text{(A-2)}$$

where ω is the frequency in radians. Substituting the eigenfunction into equation A-1 yields:

$$H(\omega) = \frac{1}{2N+1} \sum_{m=-N}^{N} e^{j(i-m)\omega} \qquad \text{(A-3)}$$

where $H(\omega)$ is the transfer function. Note that equation A-3 looks just like the discrete Fourier transform with an input function of 1. Expand the summation:

$$H(\omega) = \frac{1}{2N+1}\left(e^{-jN\omega}+e^{-j(N-1)\omega}+e^{-j(N-2)\omega}+\ldots+1+\ldots+e^{j(N-1)\omega}+e^{jN\omega}\right) \qquad \text{(A-4)}$$

But the exponential terms may be expressed as sines and cosines as follows:

$$e^{-j(N)\omega} = \cos(-N\omega) + j\sin(-N\omega)$$

$$e^{-j(N-1)\omega} = \cos\left(-(N-1)\omega\right) + j\sin\left(-(N-1)\omega\right)$$

$$e^{-j(N-2)\omega} = \cos\left(-(N-2)\omega\right) + j\sin\left(-(N-2)\omega\right)$$

$$\cdots$$

$$e^{j(N-1)\omega} = \cos\left((N-1)\omega\right) + j\sin\left((N-1)\omega\right)$$

$$e^{jN\omega} = \cos N\omega + j\sin N\omega$$

Substituting these identities into equation A-4 and simplifying yields the transfer function for the moving average filter:

$$H(\omega) = \frac{\sin(N+\frac{1}{2})\omega}{(2N+1)\sin\frac{\omega}{2}} \qquad \text{(A-5)}$$

Using equation A-5 and the identity:

$$\omega = 2\pi\left(\frac{f}{f_s}\right) \qquad \text{(A-6)}$$

where f is the frequency in Hertz and f_s is the sample rate, the frequency response characteristics of the moving average filter may be computed and plotted for different window sizes (L = 2N + 1). Figure A-1 presents the transfer function of the moving average filter for several different window sizes at frequencies up to the Nyquist frequency (see section III.B.2.). The cutoff frequency of each filter is defined as the frequency where the magnitude has decreased to 0.7071 (3 dB) of its initial value. Table IV-1 presents the cutoff frequencies for many different window sizes. These were determined by solving the following equation for f_c, the cutoff frequency:

$$0.7071 = \frac{\sin\left[(N+\frac{1}{2})2\pi\left(\frac{f_c}{f_s}\right)\right]}{(2N+1)\sin\pi\left(\frac{f_c}{f_s}\right)} \qquad \text{(A-7)}$$

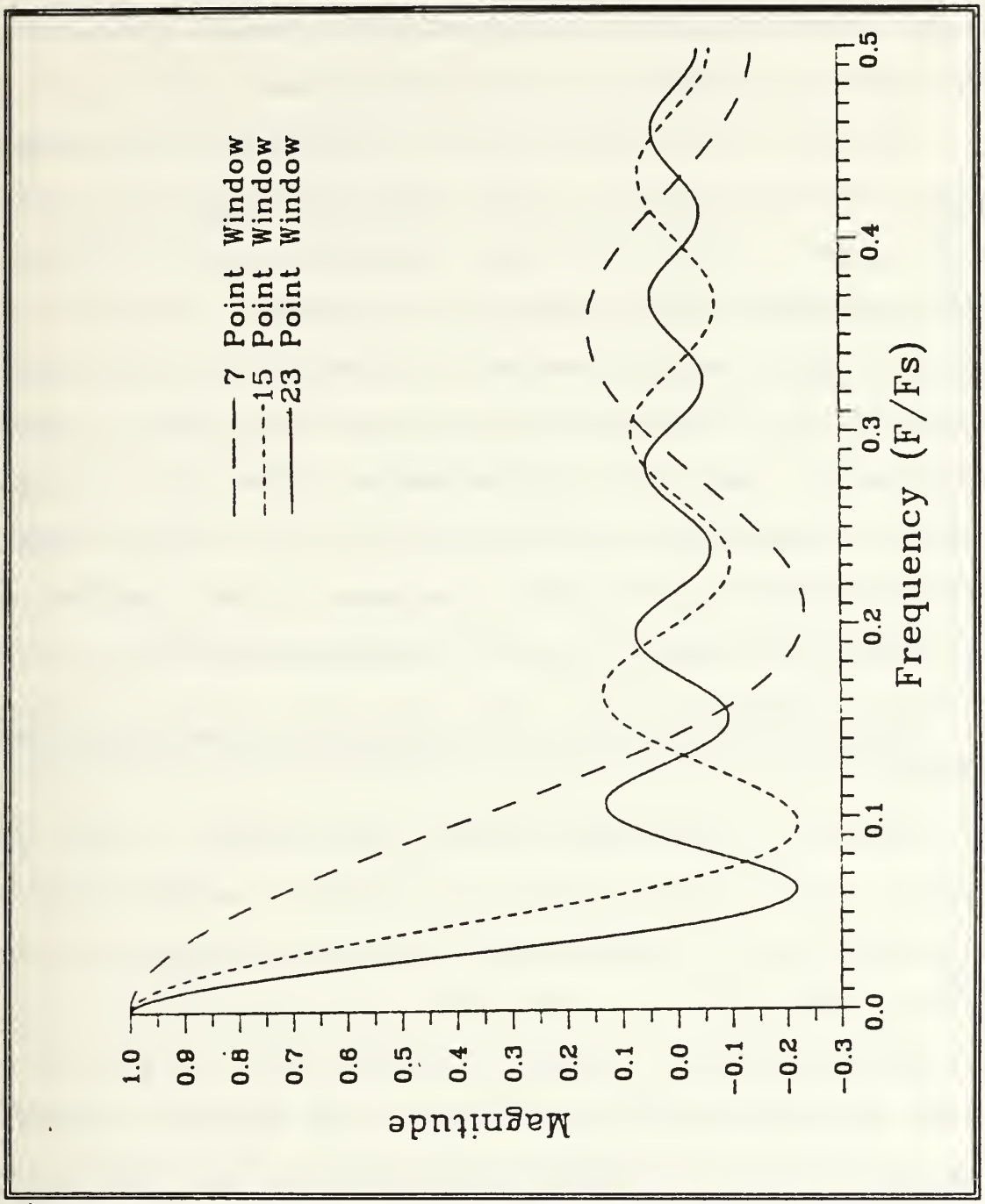

Figure A-1. TRANSFER FUNCTION OF THE MOVING AVERAGE SMOOTHING FILTER FOR THREE DIFFERENT SMOOTHING WINDOW SIZES

Program FCMA is provided in Appendix B to compute the solution for different sample rates and window sizes.

Observe the large oscillations or ripples in the frequency response characteristics of the moving average filter. These ripples are known as the Gibbs' phenomenon and are caused by the truncated Fourier series of equation A-3 [Ref. 9]. Truncating the Fourier series is necessary for a finite time series. The Gibbs' phenomenon causes the filter transfer function to yield a poor approximation of the ideal low pass filter presented in Figure IV-9. Finding better approximations to the ideal low pass filter provides the motivation for examining other smoothing algorithms.

B. FREQUENCY RESPONSE OF THE FREQUENCY DOMAIN SMOOTHING FILTER

Instead of operating on the time series in the time domain, a more natural method to accomplish low pass filtering (smoothing) is to transform the time series into the frequency domain using the DFT. Then the input time series (now a "frequency series") is simply multiplied by a low pass filter whose transfer function is defined in the frequency domain to possess the desired frequency response filter characteristics. Transfer functions may be defined which minimize or eliminate the Gibbs' phenomenon ripple effect and better approximate the ideal low pass filter.

One such transfer function, used in the frequency domain smoothing filter described in Chapter IV, is a parabolically shaped filter defined by:

$$H\left(\frac{f}{f_s}\right) = MAX\left[1-\left(\frac{L}{M}-J\right)^2, 0\right] \quad J=0,1,2,\ldots,M \tag{A-8}$$

where L is the "smoothing window" size which determines the filter cutoff frequency, M is the (integer power of two) number of data points in the time series, and J is the frequency bin index running from 0 to M. Figure A-2 summarizes the frequency response characteristics of this low pass filter. The plot presents J versus magnitude for several different smoothing window sizes. The window size is parameterized as L/M since the actual frequency response characteristics are a function of the number of points in the time series. The actual frequency associated with the index J is given by:

$$f = f_s\left(\frac{J}{M}\right) \tag{A-9}$$

As before, the cutoff frequency is defined as the frequency at which the magnitude has attenuated to 0.7071 of the initial value. Combining equations A-7 through A-8, the cutoff frequency may be determined by solving the equation:

$$0.7071 = 1-\left[L\left(\frac{f_c}{f_s}\right)\right]^2 \tag{A-10}$$

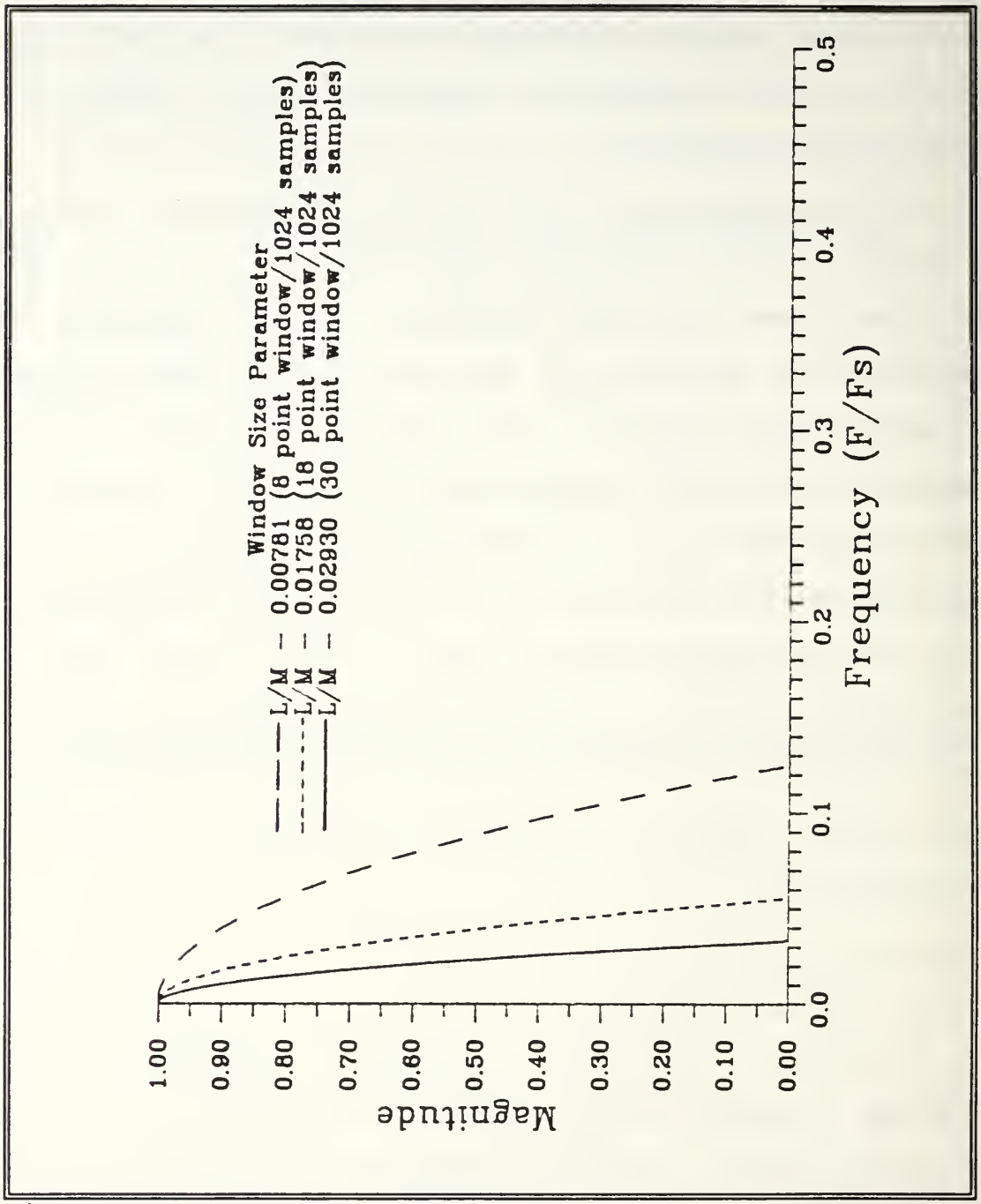

Figure A-2. TRANSFER FUNCTION OF THE FREQUENCY DOMAIN SMOOTHING FILTER FOR THREE DIFFERENT SMOOTHING WINDOW SIZES.

Which yields:

$$f_c = \sqrt{0.2929\left(\frac{f_s}{L}\right)^2} \qquad \text{(A-11)}$$

Table IV-2 was generated using equation A-11.

C. COMPARISON OF THE TRANSFER FUNCTIONS

Figure A-3 presents a comparison of the transfer functions of the moving average smoothing filter and the frequency domain smoothing filter. Figure A-3 is presented for a sample rate of 15,000 Hz and a time series length of 1024 points. The smoothing window widths were selected such that the cutoff frequencies of the two filters are nearly equal. For the moving average filter, f_c = 320 Hz, corresponding to a window size of 21 points. For the frequency domain filter, f_c = 325 Hz, corresponding to a 25 point smoothing window. The two transfer functions are nearly equal at frequencies up to the cutoff frequency. The frequency domain filter falls off more rapidly and does not oscillate. Therefore, the frequency domain smoothing filter provides a better approximation to the ideal low pass filter.

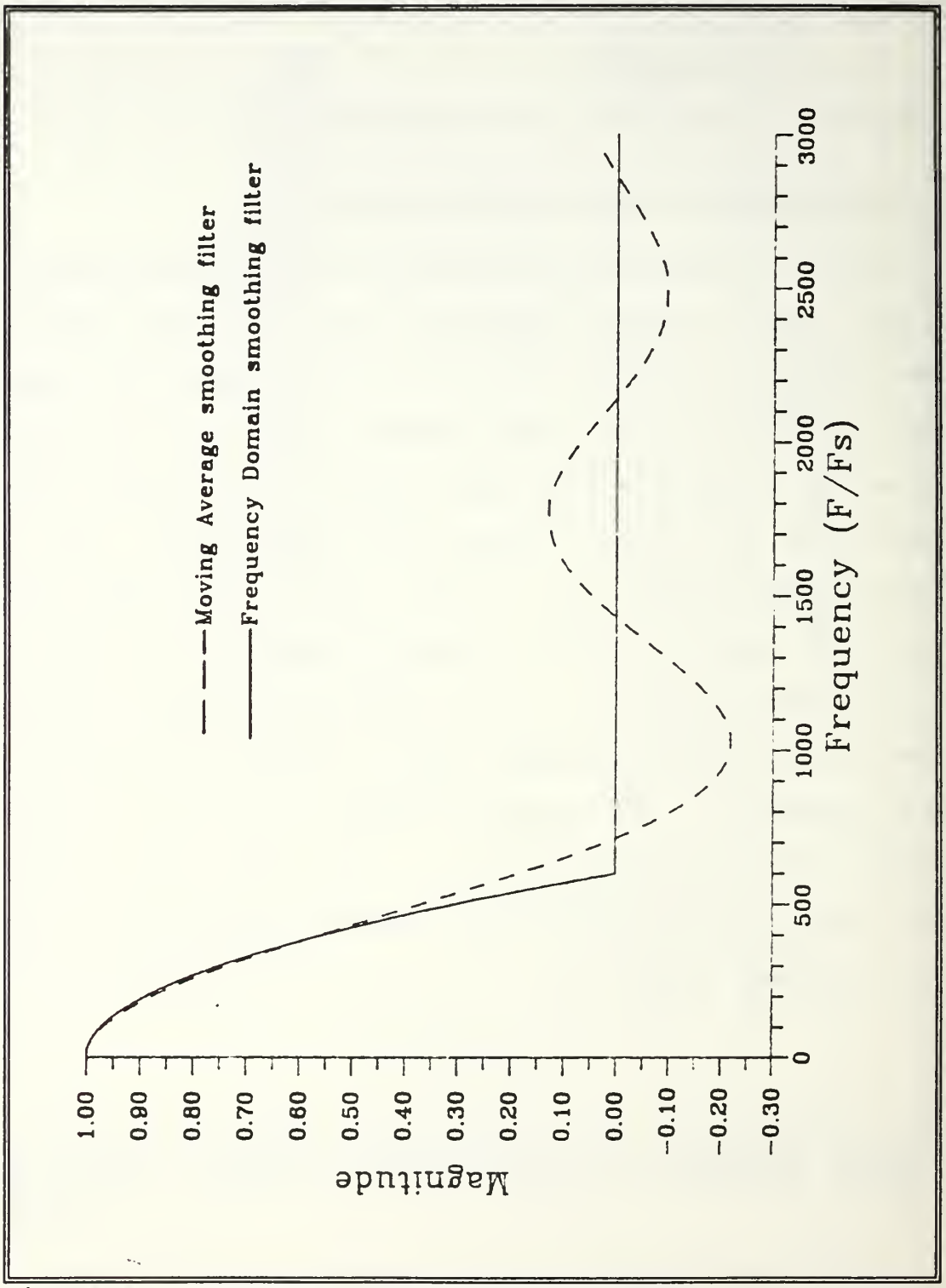

Figure A-3. COMPARISON OF FREQUENCY DOMAIN SMOOTHING (25 POINT WINDOW, Fc - 335 HZ) AND MOVING AVERAGE SMOOTHING (21 POINT WINDOW, Fc - 320 hZ). SAMPLE RATE - 15,000 HZ, RECORD LENGTH - 1024 SAMPLES.

APPENDIX B

SOURCE CODE AND USER'S GUIDES FOR FORTRAN PROGRAMS

A. GENERAL INFORMATION

This section provides general information and guidance which applies to all of the programs presented in this Appendix. Subsequent sections present user documentation specific to each individual program.

The data acquisition system planned for the tests described in Chapter II is an IBM PC/AT (R) clone microcomputer equipped with an analog-to-digital conversion board and an 80287 math co-processor. The data will be digitized and recorded on the hard disk in real time. Data analysis using the programs described in this thesis will be performed subsequent to the tests. These programs are intended to run on the same hardware setup just described.

All of the programs presented in this thesis were written and compiled with Microsoft (R) FORTRAN version 4.1 operating under Microsoft's MS DOS (R) version 3.2 operating system. Microsoft's (R) FORTRAN compiler implements the full ANSI 77 standard for the FORTRAN programming language. This version of FORTRAN also includes many extensions to ANSI 77 standard FORTRAN. Compiler options were selected to take full

advantage of the 80287 math co-processor. All program and data file names follow the MS DOS (R) standard format for file names.

The programs presented in this Appendix are all heavily commented and documented to allow the user to follow the logic and make modifications as necessary. Other individual applications will likely have different input file formats. So, the input file "read" statements are most likely to need modification. Most of the programs presented here have integrated the input file "read" statements into the algorithm which separates the input velocity data into an ensemble of pulses. This algorithm will be described first and applies to all programs which operate on the ensemble of pulses. (Programs PSD, CORREX, CORFFT and MEMPSD do not operate on an ensemble.)

The input file is expected to be an ASCII sequential file with alternating values of two parameters (either in a single column or two adjacent columns). The two parameters are expected to be velocity (U) and the propeller trigger (PROP), in that order. The propeller trigger signal provides a sharp spike at each propeller blade passage and is used to separated the velocity data into an ensemble. The program reads the velocity and prop trigger data until the first trigger spike is encountered. This is the read statement that should be

modified to handle other input file formats. The first trigger spike exceeding a value of one (volt) initiates the beginning of the first blade wake passage. The velocity samples are copied into the first column of array ENSMBL as they are read from the input file. Upon encountering the next trigger spike, the velocity samples are copied into the beginning of the next column. In this manner, array ENSMBL is filled with velocity samples such that each column represents a blade wake passage. Data is read until 50 blade wake passages have been copied into array ENSMBL or until an end-of-file (EOF) is encountered. (The last partial pulse which was being read when the EOF was encountered is deleted since it is incomplete.) The length of the shortest column is saved (LASTI) so that the entire array may be truncated to the length of the shortest blade wake passage.

The length of each column is displayed on the screen as the program runs. The user should watch these values carefully to insure that the columns are all about the same length. Any significant difference in pulse length (more than about five to ten samples) is a possible indication of a problem with the propeller trigger signal. Erroneous results may be obtained if there are significant discrepancies in the length of the blade wake passages.

B. USER'S GUIDE FOR PROGRAM ENSEMBL

1. Input Instructions

- Insure that the executable file for program ENSEMBL (ENSEMBL.EXE) and the input data file are in the same subdirectory of a disk drive with sufficient space remaining to write the output file.
- Initiate program ENSEMBL (type "ENSEMBL" at the DOS prompt).
- The program will prompt the user for the name of the input data file. Enter the full name of the input data file including the extension.
- The program will prompt the user for the name of the output file which will be created by the program. Enter the desired file name. Do not use the name of an existing file or the program will terminate.
- The program will then read the input data file, compute the ensemble average and write the output data file. Remember to observe the column lengths of the array ENSMBL to insure that the ensemble is being constructed properly.

2. Output Description

Program ENSEMBL computes the ensemble average and writes an output file consisting of three columns. The first column is ensemble sample number (or row number, running from 1 to LASTI). The second column is the ensemble average values. The third column is the ensemble variance. The following pages present the source code for program ENSEMBL.

```
      PROGRAM ENSEMBL
************************************************************************
*   Program to perform ensemble averaging on an ensemble of unsteady
*   turbulent boundary layer velocity measurements such as occurs
*   in the boundary layer of an airfoil immersed in a propeller slipstream
*
*   Input data file is assumed to be in output format of the A/D conversion
*   system. For this application the input file is a sequential
*   file with alternating values of velocity (U) and propeller pulse
*   (PROP), starting with velocity.
*
*   The program first separates the single time history input file into an
*   ensemble of pulses using the value of the propeller trigger (PROP)
*   as the marker for the beginning of a pulse. The program also calculates
*   statistical variance and a "pseudo" turbulence intensity for the ensemble.
*
*
*   Key variables are defined below:
*
*  ENSMBL = The primary data array which contains up to 50 pulses (columns)
*           each of length up to 2048 points. The pulses are synchronized in
*           time (an ensemble) such that the beginning of each column (i.e.
*           the first row) represents the beginning of each pulse.
*  U = The measured velocity as read from the input file into array ENSMBL.
*  PROP = The propeller pulse timing marker signal. Also read from the input.
*  IPLS = Number of pulses to be read from the input file.
*  LASTI = Number of points in the columns of ENSMBL. (ie. rows in ENSMBL)
*  NPLS = (J-1) = Final total number of pulses (columns in ENSMBL).
*  AVG = The ensemble average of U. (Output)
*  VAR = The ensemble statistical variance of U. (Output)
*  TURBIN = The "pseudo" ensemble turbulence intensity of U. (Not based
*           on local smoothed velocity but on the overall ensemble mean
*           velocity. Really a variance.) Output.
*  UINF = Free stream velocity external to the boundary layer.
*
*                  Programmed by: Donald K. Johnson
*                           July 1988,NPS
*
*
************************************************************************
      DIMENSION ENSMBL(2048,50),AVG(2048),VAR(2048),TURBIN(2048)
      INTEGER I,IPLS,LASTI,J,K,L,NPLS
      CHARACTER*12 INFILE,OUTFILE
*
      PRINT'(/A/)',' Program to compute pure ensemble average:'
*
      PRINT '(/A\)',' ENTER THE INPUT DATA FILE NAME: '
      READ (*,'(A)') INFILE
*
*     PRINT '(/A\)',' ENTER THE NUMBER OF PROP PULSES YOU WANT: '
*     READ (*,*) IPLS
      IPLS=50
*
      PRINT '(/A\)',' ENTER THE OUTPUT DATA FILE NAME: '
      READ (*,'(A)') OUTFILE
```

```
*
*     pseudo turbulence intensity calcs are currently de-activated ***
*     PRINT '(/A\)',' ENTER U-edge (REFERENCE VELOCITY FOR TURB.INTENSITY
*    1 NORMALIZATION): '
*     READ(*,*) UINF
*     UINF=UINF*1.
      UINF=1.
*
      OPEN(10,FILE=INFILE,STATUS='OLD')
      OPEN(11,FILE=OUTFILE,STATUS='NEW')
*
      LASTI=2048
      J=1
      I=0
      PRINT'(/A/)',' READING INPUT FILE....'
      PRINT'(/A/)',' WARNING, Watch the number of points in each column
     1... they should all be about the same length.'
*
 10   CONTINUE
***     Read the data into array "ENSMBL", each column is a pulse ***
        READ (10,*,END=20,ERR=20) U,PROP
        I=I+1
***     But skip the first partial pulse ******
        IF (J .GT. 1) ENSMBL(I,J-1)=U
***     If you have a new pulse, start a new column:
*       a new pulse is defined by PROP being greater than 1. volt
        IF(I .GE. 20 .AND. PROP .GT. 1.) THEN
            IF (J .EQ. 1) THEN
               WRITE(*,*)'Skipped first',I,' points (not a full pulse).'
            ELSE
               WRITE(*,*)'No. of points in column',J-1,' is',I
            ENDIF
***         Find the shortest column ***
            IF(I .LT. LASTI  .AND.  J .GT. 1) LASTI=I
***         Quit loop if you have enough pulses ***
            IF (J-1 .GE. IPLS) GO TO 28
            J=J+1
            I=0
        ENDIF
      GO TO 10
 20   CONTINUE
         WRITE(*,*)'End of File encountered - deleting last partial colu
     1mn....'
         J=J-1
 28   CONTINUE
      XN=FLOAT(J-1)
      NPLS=J-1
*
      WRITE(*,*)'The final number of pulses is',NPLS,' each of length',
     1 LASTI
*** Compute the ensemble mean, variance and "pseudo" turbulence intensity
*   based on user input value of V - infinity (free stream velocity).
C
      DO 22 I=1,LASTI
        SUM=0.
```

```
        SUMSQ=0.
        DO 30 L=1,NPLS
***        first compute the sum ***
           SUM=SUM+ENSMBL(I,L)
           SUMSQ=SUMSQ+ENSMBL(I,L)**2
  30    CONTINUE
***     then divide by the number of points
        AVG(I)=SUM/XN
        VAR(I)=(SUMSQ-XN*AVG(I)**2)/(XN-1.)
        TURBIN(I)=(((SUMSQ-XN*AVG(I)**2)/XN)**.5)/UINF
  22  CONTINUE
*** Write the output ******
       DO 40 I=1,LASTI
         XI=FLOAT(I)
***      note: the pseudo turbulence intensity is currently not output
         WRITE(11,200) XI,AVG(I),VAR(I)
 200     FORMAT(F5.0,F8.1,F8.1)
  40  CONTINUE
      CLOSE(10)
      CLOSE (11)
      END
```

C. USER'S GUIDE FOR PROGRAM MOVAVE

1. Input Instructions

- Insure that the executable file for program MOVAVE (MOVAVE.EXE) and the inpute data file are in the same subdirectory of a disk drive with sufficient space remaining to write the output file.

- Initiate program MOVAVE (type "MOVAVE" at the DOS prompt).

- The program will prompt the user for the name of the input data file. Enter the full name of the input data file including the extension.

- The program will prompt the user for the name of the output file which will be created by the program. Enter the desired file name. Do not use the name of an existing file or the program will terminate.

- The program will prompt the user for the moving average window size. The window size specifies the number of samples which are time averaged to produce an interim "local" mean or "local" smoothed velocity which is then ensemble averaged. The window size should be an odd number. Table IV-1 in Chapter IV presents some examples of the relationship between moving average window size and its associated low pass filter cutoff frequency. Other cutoff frequencies may be determined by solving equation A-7 in Appendix A. Program FCMA is provided (immediately after the source code for program MOVAVE) to solve equation A-7.

- The program will then read the input data file, compute the non-stationary mean and write the output data file. Remember to observe the column lengths of the array ensembl to insure that the ensemble is being constructed properly.

2. Output Description

Program MOVAVE computes the non-stationary mean and writes an output file consisting of two columns. The first column is ensemble sample number (or row number, running from

1 to LASTI). Knowing the sample rate, the time scale may be inferred from sample number. The second column contains the non-stationary mean values. The following pages present the source code for program ENSEMBL.

```
$DEBUG
      PROGRAM MA
C*************************************************************************
*    Program to perform simple moving average smoothing on an ensemble of
*    propeller pulses then ensemble average the smoothed data. Note that
*    the moving average operation sets the first (L-1)/2 points and the last
*    (L-1)/2 points  of the mean velocity output to zero (where L is the
*    smoothing window size). The relatioship between smoothing window
*    size and cutoff frequency is given by:
*
*     SIN(((L-1)/2 + .5)*2*PI*FC/FS) / L*SIN(PI*FC/FS) = 0.7071
*
*    where FC is the cutoff frequency, FS is the sample rate and L is the
*    odd numbered window size. The table below gives some examples:
*
*                        L                    FC  (FS=30000 Hz)
*                       ___                  _____
*                        3                   4660 Hz
*                       11                   1220
*                       17                    790
*                       21                    640
*                       35                    380
*
*    Input data file is assumed to be in output format of the D/A
*    conversion system, i.e. a sequential file with alternating values
*    of velocity (U) and propeller pulse (PROP), starting with velocity.
*    The program first separates the single time history file into an
*    ensemble of pulses using the value of the propeller trigger (PROP)
*    as the marker for the beginning of a pulse.
*
*    Key variables are defined below:
*
*   ENSMBL = The primary data array which contains up to 50 pulses (columns)
*            each of length up to 2048 points. The pulses are synchronized in
*            time (an ensemble) such that the beginning of each column (i.e.
*            the first row) represents the beginning of each pulse.
*   U = The measured velocity as read from the input file into array ENSMBL.
*   PROP = The propeller pulse timing marker signal. Also read from the input.
*   IPLS = Number of pulses to be read from the input file.
*   LASTI = Number of points in the columns of ENSMBL. (ie. rows in ENSMBL)
*   NPLS = (J-1) = Final total number of pulses (columns in ENSMBL).
*   L = The width of the moving average window  (should be an odd number).
*   A(I) = The current column of ENSMBL to be smoothed using subroutine MOVAVE
*   B(I) = The current smoothed column of ENSMBL.
*   USMTH(I) = The ensemble average of all the smoothed propeller pulses.
*
*
*                  Programmed by: Donald K. Johnson
*                         AUGUST,1988,NPS
*
*************************************************************************
      PARAMETER (MMAX=2048)
      DIMENSION ENSMBL(2048,50),A(MMAX),B(MMAX),USMUTH(MMAX)
      INTEGER L,NPTS,ILAST,NPLS,IPLS
      CHARACTER*12 INFILE,OUTFILE
```

```
*
      PRINT'(/A/)',' Moving average smoothing and ensemble averaging.'
*
      PRINT '(/A\)',' ENTER THE INPUT DATA FILE NAME: '
      READ (*,'(A)') INFILE
*
      PRINT '(/A\)',' ENTER THE OUTPUT FILE NAME: '
      READ (*,'(A)') OUTFILE
*
      PRINT '(/A\)',' ENTER THE MOVING AVERAGE WINDOW SIZE" (AN ODD NUMB
     1ER): '
      READ (*,*) L
*
      OPEN(UNIT=10,FILE=INFILE,STATUS='OLD')
      OPEN(UNIT=11,FILE=OUTFILE,STATUS='NEW')
*
***   separate the time history data into an ensemble of pulses ***
      LASTI=2048
      IPLS=50
      J=1
      I=0
      PRINT'(/A/)',' READING INPUT FILE....'
      PRINT'(/A/)',' WARNING, Watch the number of points in each column
     1... they should all be about the same length.'
*
 10   CONTINUE
***     Read the data into array "ENSMBL", each column is a pulse ***
        READ (10,*,END=20,ERR=20) U,PROP
        I=I+1
***     But skip the first partial pulse ******
        IF (J .GT. 1) ENSMBL(I,J-1)=U
***     If you have a new pulse, start a new column. ***
*       Note, prop trigger is set to 1. volt to start a new pulse ***
        IF(I .GE. 20 .AND. PROP .GT. 1.) THEN
            IF (J .EQ. 1) THEN
               WRITE(*,*)'Skipped first',I,' points (not a full pulse).'
            ELSE
               WRITE(*,*)'No. of points in column',J-1,' is',I
            ENDIF
***         Find the shortest column ***
            IF(I .LT. LASTI  .AND.  J .GT. 1) LASTI=I
***         Quit loop if you have enough pulses ***
            IF (J-1 .GE. IPLS) GO TO 28
            J=J+1
            I=0
        ENDIF
      GO TO 10
 20   CONTINUE
        WRITE(*,*)'End of File encountered - deleting last partial colu
     1mn....'
        J=J-1
 28   CONTINUE
      XN=FLOAT(J-1)
      NPLS=J-1
*
```

```
      WRITE(*,*)'The final number of pulses is',NPLS,' each of length',
     1 LASTI
*
***   apply the moving average window to each column (pulse) in array ENSMBL
*     writing the averaged value back into array ENSMBL at the current point.
*
      PRINT '(/A/)',' WORKING .....'
***   start the pulse loop ***
      DO 25 K=1,NPLS
***      and start the time loop ***
         DO 15 I=1,LASTI
***         copy the current (Kth) pulse into A for call to subroutine ***
            A(I)=ENSMBL(I,K)
 15      CONTINUE
***      smooth the Kth pulse ***
         CALL MOVAVE(A,LASTI,L,B)
*
***      write the smoothed values of velocity back into ENSMBL
         DO 16 I=1,LASTI
            ENSMBL(I,K)=B(I)
 16      CONTINUE
 25   CONTINUE
*
***   now ensemble average (across rows) to get the final mean velocity
***   start with a time loop ***
      DO 50 I=1,LASTI
***   now sum across each row ***
      SUMU=0.
         DO 60 K=1,NPLS
           SUMU=SUMU+ENSMBL(I,K)
 60      CONTINUE
*** compute the ensemble average ***
         USMUTH(I)=SUMU/NPLS
 50   CONTINUE
***   write the output ***
      DO 70 I=1,LASTI
         XI=FLOAT(I)
         WRITE(11,100) XI,USMUTH(I)
 100     FORMAT(1X,F5.0,1X,F10.2)
 70   CONTINUE
      CLOSE(10)
      CLOSE(11)
      END
C*************************************************************************
      SUBROUTINE MOVAVE(A,N,L,B)
*
*   Program performs a simple moving average of a time series to determine a
*   local mean (smoothed) velocity. Note that the first L/2 values and the
*   last L/2 values of the input time series are set to zero. Briefly, the
*   moving average accomplishes a crude low pass filter type of smoothing
*   on the input time series as follows. Starting with the first L (usually
*   odd) points in the time series, a time average is computed for the L
*   points. This average value becomes the smoothed velocity at the center
*   of the L points, point (L-1)/2 for odd L. Then the moving average window
*   is moved over one point and a new average is computed for the window. This
```

```
*   is repeated until the Lth point of the window reaches the end of the time
*   series.
*
*   The key variables are defined below:
*
*   A = A column array containing the time series to be averaged.
*   N = The number of values in A.
*   L = The size of the averaging window (an odd number is best because
*       then the average of the small time segment is centered on the
*       mid-point of the segment). Note, a larger window size corresponds to
*       a lower low pass filter cutoff frequency. See the table below for the
*       relation between window size and cutoff frequency.
*   B = A column array containing the smoothed (averaged) time series values.
*
      DIMENSION A(1),B(1)
      AL=L
      MH=L/2
***   zero out the answer array ***
      DO 18 I=1,N
   18    B(I)=0.
      K=L-1
***   if L is .GT. N then quit ***
      IF (N-L) 1,2,2
    2 NN=N-K
***   start the main averaging loop, working on L/2 points on either
*     side of the KK th data point. (KK is calculated below)
      DO 4 J=1,NN
         JN=J+K
***      reset the accumulator ***
         S=0.
***      sum across L points centered on point KK.
         DO 3 I=J,JN
            S=S+A(I)
    3    CONTINUE
***      now divide by L to get the average of segment centered on point KK.
         KK=J+MH
         B(KK)=S/AL
    4 CONTINUE
***   if L is even then take the average of two adjacent terms to get the value
*     of the average velocity centered on the KK th point.
      IF (L-(L/2)*2) 1,6,1
    6 MS=MH+1
      MD=MH+NN-1
      DO 7 I=MS,MD
         TEMP=(B(I)+B(I+1))/2.
         B(I)=TEMP
    7 CONTINUE
    1 RETURN
      END
```

```
$DEBUG
      PROGRAM FCMA
*   Computes the relationship between the moving average smoothing window
*   size (L, where L=2*N + 1), the sample rate (FS), and the frequency (F),
*   so that the low pass moving average smoothing filter cutoff frequency
*   may be determined. Cutoff frequency is defined as the frequency where
*   the magnitude is -3 dB (a factor of 0.7071) down from the unfiltered
*   magnitude.
*
      PI=4.*ATAN(1.)
      PRINT '(/A\)',' ENTER SAMPLE RATE: '
      READ(*,*) FS
*
***   start the window size loop (window size, L = 2N+1)  ***
*     this loop will compute the cutoff frequency for all window sizes
*     between 3 and 61.
      DO 10 N=1,30
***     start with 5 Hz. frequency and step up in 5 Hz. steps ***
        F=5.
        KNT=0
 20      CONTINUE
***      compute the magnitude ***
         XN=FLOAT(N)
         X1=SIN((2*PI*F/FS)*(XN+.5))
         X2=(2.*XN+1.)*SIN(PI*F/FS)
         H=X1/X2
         L=2*N+1
***      if the magnitude is near the cutoff point then print results. ***
*         the correct cutoff frequency is the one where the magnitude is
*         less than or equal to 0.7071
         IF(H .LT. 0.708) THEN
            WRITE(*,100) L,F,H
 100        FORMAT(1X,I5,F8.0,F10.6)
            KNT=KNT+1
***        after writing results for 3 frequencies, skip to next window size
            IF(KNT .GT. 3) GO TO 10
         ENDIF
***      increment frequency ***
         F=F+5.
         IF (F .LT. FS) GO TO 20
 10   CONTINUE
      END
```

D. USER'S GUIDE FOR PROGRAM SMUMEAN

1. Input Instructions

- Insure that the executable file for program SMUMEAN (SMUMEAN.EXE) and the input data file are in the same subdirectory of a disk drive with sufficient space remaining to write the output file.

- Initiate program SMUMEAN (type "SMUMEAN" at the DOS prompt).

- The program will prompt the user for the name of the input data file. Enter the full name of the input data file including the extension.

- The program will prompt the user for the name of the output file which will be created by the program. Enter the desired file name. do not use the name of an existing file or the program will terminate.

- The program will prompt the user for the FFT smoothing window size. The window size specifies the cutoff frequency of the low pass frequency domain filter. The low pass filtered data produces an interim "local" mean or local smoothed velocity which is then ensemble averaged. Table IV-2 in Chapter IV presents some examples of the relationship between moving average window size and its associated low pass filter cutoff frequency. Other cutoff frequencies may be calculated using equation A-11 in Appendix A.

- The program will then read the input data file, compute the non-stationary mean and write the output data file. Remember to observe the column lengths of the array ENSMBL to insure that the ensemble is being constructed properly.

2. Output Description

Program SMUMEAN computes the non-stationary mean and writes an output file consisting of two columns. The first column is ensemble sample number (or row number, running from 1 to LASTI). Knowing the sample rate, the time scale may be

inferred from sample number. The second column contains the non-stationary mean values. The following pages present the source code for program ENSEMBL.

```
      PROGRAM SMUMEAN
*************************************************************************
*   Program to perform low-pass FFT digital filter smoothing and then
*   ensemble averaging on the smoothed turbulent boundary layer velocity
*   data. In other words, this program computes the time varying mean of
*   a non-stationary time series.
*
*   The cutoff frequency of the low pass filter is specified by the
*   size of the smoothing window, PTS, the sample rate and the number
*   of points to be smoothed, M. See the table below:
*
*         Window Size    -3 dB Cutoff Frequency (FS=30000,M=2048)
*              2                 8120
*              5                 3250
*             10                 1625
*             15                 1080
*             20                  810
*             25                  650
*             30                  540
*             35                  465
*             40                  405
*             50                  325
*
*   For other sample rates and window sizes, the cutoff frequency may
*   be determined from the equation:
*
*                FC = (0.29289 * (FS/PTS)**2)**0.5
*
*   where FC is the cutoff frequency, and FS is the sample rate.
*   Note, the input array to the smoothing subroutine does not have to be
*   an integer power of two because the subroutine zero pads as necessary
*   to the next highest integer power of two greater than the length of
*   the data vector (in this case the length of the columns of ENSMBL,
*   which is LASTI.*
*
*   The input data file is assumed to be in output format of the A/D
*   conversion system, ie. a sequential file with alternating
*   values of velocity (U) and propellor pulse (PROP), starting with
*   velocity. Smoothing is accomplished with subroutine SMOOFT which
*   performs frequency domain digital (low pass) filtering of
*   the velocity signal. Program  calculates ensemble mean velocity
*   from the smoothed velocity values. Variance and turbulence
*   intensity are computed in program TURBIN. Array size reqirements
*   prevent doing both in one pass through the data.
*
*   Key variables are defined below:
*
*  ENSMBL = The primary data array which contains up to 50 pulses (columns)
*           each of length up to 2048 points. The pulses are synchronized in
*           time (an ensemble) such that each point in a pulse "lines up"
*           with the others.
*  U = The measured velocity as read from the input file into array ENSMBL.
*  PROP = The propeller pulse timing marker signal. Also read from the input.
*  IPLS = Number of pulses to be read from the input file.
*  LASTI = Number of points in the columns of ENSMBL. (ie. rows in ENSMBL)
```

```
*  NPLS = (J-1) = Total number of pulses (columns in ENSMBL).
*  AVG = The ensemble averaged then FFT smoothed value of U. (Output)
*
*                  Programmed by: Donald K. Johnson
*                          Sept. 1988,NPS
*
*
*
*************************************************************************
      DIMENSION ENSMBL(2048,50),AVG(2048)
      DIMENSION UCOL(2048)
      INTEGER I,IPLS,LASTI,J,K,L
      CHARACTER*12 INFILE,OUTFILE
C
      PRINT '(/A/)',' Low-Pass Filter Smoothing then Ensemble Average'
C
      PRINT '(/A\)',' ENTER THE INPUT DATA FILE NAME: '
      READ (*,'(A)') INFILE
C
C     PRINT '(/A\)',' ENTER THE NUMBER OF PROP PULSES YOU WANT: '
C     READ (*,*) IPLS
      IPLS=50
C
      PRINT '(/A\)',' ENTER THE OUTPUT DATA FILE NAME: '
      READ (*,'(A)') OUTFILE
C
      PRINT '(/A\)',' ENTER THE FFT SMOOTHING WINDOW SIZE: '
      READ(*,*) PTS
C
      OPEN(10,FILE=INFILE,STATUS='OLD',FORM='FORMATTED')
      OPEN(11,FILE=OUTFILE,STATUS='NEW')
C
      LASTI=2048
      J=1
      I=0
      PRINT'(/A/)',' Smoothing then ensemble averaging:'
      PRINT'(/A/)',' Reading input file...'
C
 10   CONTINUE
***     Read the data into array "ENSMBL", each column is a pulse ***
        READ (10,*,END=20,ERR=20) U,PROP
        I=I+1
***     But skip the first partial pulse ******
        IF (J .GT. 1) ENSMBL(I,J-1)=U
***     If you have a new pulse, start a new column
*       prop trigger currently set to 1. volt ***
        IF(I .GE. 20 .AND. PROP .GT. 1.) THEN
            IF (J .EQ. 1) THEN
               WRITE(*,*)'Skipped first',I,' points (not a full pulse).'
            ELSE
               WRITE(*,*)'No. of points in column',J-1,' is',I
            ENDIF
***         Find the shortest column ***
            IF(I .LT. LASTI  .AND.  J .GT. 1) LASTI=I
            IF (J-1 .GE. IPLS) GO TO 28
```

```
            J=J+1
            I=0
          ENDIF
       GO TO 10
 20    CONTINUE
          PRINT '(/A/)',' End of File or read error encountered - deletin
     1g last partial column....'
          J=J-1
 28    CONTINUE
       XN=FLOAT(J-1)
       NPLS=J-1
C
        WRITE(*,*)  ' The final number of pulses is',J-1,'    each of len
     1gth',LASTI
C
*** Perform the FFT smoothing on each column of the array ENSMBL ***
       DO 300 L=1,NPLS
       WRITE(*,*)'Smoothing column',L
          DO 310 I=1,LASTI
             UCOL(I)=ENSMBL(I,L)
 310      CONTINUE
          CALL SMOOFT(UCOL,LASTI,PTS)
***       Write the smoothed velocity back into ENSMBL ***
          DO 320 I=1,LASTI
             ENSMBL(I,L)=UCOL(I)
 320      CONTINUE
 300   CONTINUE
C
*** Compute the ensemble mean ***
C
       DO 22 I=1,LASTI
         SUM=0.
         SUMSQ=0.
         DO 30 L=1,NPLS
            SUM=SUM+ENSMBL(I,L)
  30     CONTINUE
         AVG(I)=SUM/XN
  22   CONTINUE
*** Write the output ******
        DO 40 I=1,LASTI
          XI=FLOAT(I)
          WRITE(11,200) XI,AVG(I)
 200      FORMAT(F5.0,F8.1)
  40   CONTINUE
       CLOSE(10)
       CLOSE (11)
       END
C*********************************************************************
       SUBROUTINE SMOOFT(Y,N,PTS)
C---------------------------------------------------------------------
C  Y IS THE DATA ARRAY, N IS THE LENGTH OF THE DATA ARRAY, AND PTS IS
C  THE SMOOTHING WINDOW WIDTH.
C
C  This subroutine is from the book "Numerical Reciepes" by Press,
C  Flannery, Teukolsky and Vetterling, Pub. 1986, Cambridge Univ. Press
```

```
C-----------------------------------------------------------------------
      PARAMETER(MMAX=2048)
      DIMENSION Y(MMAX)
C
C  M IS CALCULATED TO BE THE SMALLEST POWER OF 2 WHICH IS GREATER THAN
C  THE NUMBER OF POINTS N. THE REMAINDER OF THE ARRAY IS ZERO PADDED.
C
      M=2
      NMIN=N+2.*PTS
1     IF(M.LT.NMIN)THEN
        M=2*M
      GO TO 1
      ENDIF
      IF(M.GT.MMAX) PAUSE 'MMAX too small'
      CONST=(PTS/M)**2
      Y1=Y(1)
      YN=Y(N)
      RN1=1./(N-1.)
C
C  REMOVE THE LINEAR TREND
C
      DO 11 J=1,N
        Y(J)=Y(J)-RN1*(Y1*(N-J)+YN*(J-1))
11    CONTINUE
      IF(N+1.LE.M)THEN
C
C  ZERO PAD
C
        DO 12 J=N+1,M
          Y(J)=0.
12      CONTINUE
      ENDIF
C
C TRANSFORM TO FREQUENCY DOMAIN
C
      MO2=M/2
      CALL REALFT(Y,MO2,1)
      Y(1)=Y(1)/MO2
      FAC=1.
C
C     WRITE(11,*)'      BIN (I)           Y(F)'
C     DO 99 I=1,M
C       WRITE(11,*) I,Y(I)
C 99  CONTINUE
C
C  WINDOW FUNCTION -  (USES THE PARABOLIC WELCH WINDOW)
C
C     WRITE(11,*) ' WINDOW FUNCTION:'
      DO 13 J=1,MO2-1
        K=2*J+1
        IF(FAC.NE.0.)THEN
          FAC=AMAX1(0.,(1.-CONST*J**2)/MO2)
C         WRITE(11,*) K,K+1,FAC
          Y(K)=FAC*Y(K)
          Y(K+1)=FAC*Y(K+1)
```

```
        ELSE
          Y(K)=0.
          Y(K+1)=0.
        ENDIF
13    CONTINUE
C
C  HANDLES THE LAST POINT
C
      FAC=AMAX1(0.,(1.-0.25*PTS**2)/MO2)
      Y(2)=FAC*Y(2)
C     WRITE(11,*) ' WINDOW x DATA:'
C     DO 98 I=1,M
C     WRITE(11,*) I,Y(I)
C 98  CONTINUE
C
C  TRANSFORM BACK TO TIME DOMAIN
C
      CALL REALFT(Y,MO2,-1)
C
C  RESTORE LINEAR TREND
C
      DO 14 J=1,N
        Y(J)=RN1*(Y1*(N-J)+YN*(J-1))+Y(J)
14    CONTINUE
      RETURN
      END
C*********************************************************************
      SUBROUTINE REALFT(DATA,N,ISIGN)
C---------------------------------------------------------------------
C  CALCULATES THE FOURIER TRANSFORM OF A SET OF 2*N REAL VALUED DATA
C  POINTS (ARRAY 'DATA'). ISIGN IS 1 FOR THE FOURIER TRANSFORM (TIME
C  TO FREQ. DOMAIN) AND ISIGN IS -1 FOR INVERSE TRANSFORMATIOM.
C  THIS SUBROUTINE SETS UP THE DATA TO BE EFFICIENTLY TRANSFORMED BY
C  SUBROUTINE 'FOUR1'.
C
C  This subroutine is from the book "Numerical Reciepes" by Press,
C  Flannery, Teukolsky and Vetterling, Pub. 1986, Cambridge Univ. Press
C---------------------------------------------------------------------

      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION DATA(*)
      THETA=6.28318530717959D0/2.0D0/DBLE(N)
      C1=0.5
C
C  FORWARD TRANSFORM BY FOUR1
C
      IF (ISIGN.EQ.1) THEN
        C2=-0.5
        CALL FOUR1(DATA,N,+1)
      ELSE
C
C  SET UP FOR INVERSE TRANSFORM
C
        C2=0.5
        THETA=-THETA
```

```
      ENDIF
      WPR=-2.0D0*DSIN(0.5D0*THETA)**2
      WPI=DSIN(THETA)
      WR=1.0D0+WPR
      WI=WPI
      N2P3=2*N+3
      DO 11 I=2,N/2+1
        I1=2*I-1
        I2=I1+1
        I3=N2P3-I2
        I4=I3+1
        WRS=SNGL(WR)
        WIS=SNGL(WI)
        H1R=C1*(DATA(I1)+DATA(I3))
        H1I=C1*(DATA(I2)-DATA(I4))
        H2R=-C2*(DATA(I2)+DATA(I4))
        H2I=C2*(DATA(I1)-DATA(I3))
        DATA(I1)=H1R+WRS*H2R-WIS*H2I
        DATA(I2)=H1I+WRS*H2I+WIS*H2R
        DATA(I3)=H1R-WRS*H2R+WIS*H2I
        DATA(I4)=-H1I+WRS*H2I+WIS*H2R
        WTEMP=WR
        WR=WR*WPR-WI*WPI+WR
        WI=WI*WPR+WTEMP*WPI+WI
11    CONTINUE
      IF (ISIGN.EQ.1) THEN
        H1R=DATA(1)
        DATA(1)=H1R+DATA(2)
        DATA(2)=H1R-DATA(2)
      ELSE
        H1R=DATA(1)
        DATA(1)=C1*(H1R+DATA(2))
        DATA(2)=C1*(H1R-DATA(2))
C
C  INVERSE TRANSFORM
C
        CALL FOUR1(DATA,N,-1)
      ENDIF
      RETURN
      END
C*****************************************************************
      SUBROUTINE FOUR1(DATA,NN,ISIGN)
C---------------------------------------------------------------
C  COMPUTES THE COMPLEX DFT OR INVERSE DFT OF ARRAY 'DATA' OF
C  LENGTH 'NN'. ISIGN IS 1 FOR THE DFT AND -1 FOR THE INVERSE
C  DFT.
C
C  This subroutine is from the book "Numerical Reciepes" by Press,
C  Flannery, Teukolsky and Vetterling, Pub. 1986, Cambridge Univ. Press
C---------------------------------------------------------------
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION DATA(*)
      N=2*NN
      J=1
      DO 11 I=1,N,2
```

```
      IF(J.GT.I)THEN
        TEMPR=DATA(J)
        TEMPI=DATA(J+1)
        DATA(J)=DATA(I)
        DATA(J+1)=DATA(I+1)
        DATA(I)=TEMPR
        DATA(I+1)=TEMPI
      ENDIF
      M=N/2
1     IF ((M.GE.2).AND.(J.GT.M)) THEN
        J=J-M
        M=M/2
      GO TO 1
      ENDIF
      J=J+M
11    CONTINUE
      MMAX=2
2     IF (N.GT.MMAX) THEN
        ISTEP=2*MMAX
        THETA=6.28318530717959D0/(ISIGN*MMAX)
        WPR=-2.D0*DSIN(0.5D0*THETA)**2
        WPI=DSIN(THETA)
        WR=1.D0
        WI=0.D0
        DO 13 M=1,MMAX,2
          DO 12 I=M,N,ISTEP
            J=I+MMAX
            TEMPR=SNGL(WR)*DATA(J)-SNGL(WI)*DATA(J+1)
            TEMPI=SNGL(WR)*DATA(J+1)+SNGL(WI)*DATA(J)
            DATA(J)=DATA(I)-TEMPR
            DATA(J+1)=DATA(I+1)-TEMPI
            DATA(I)=DATA(I)+TEMPR
            DATA(I+1)=DATA(I+1)+TEMPI
12        CONTINUE
          WTEMP=WR
          WR=WR*WPR-WI*WPI+WR
          WI=WI*WPR+WTEMP*WPI+WI
13      CONTINUE
        MMAX=ISTEP
      GO TO 2
      ENDIF
      RETURN
      END
```

E. USER'S GUIDE FOR PROGRAM MEANSMU

1. Input Instructions

- Insure that the executable file for program MEANSMU (MEANSMU.EXE) and the input data file are in the same subdirectory of a disk drive with sufficient space remaining to write the output file.

- Initiate program MEANSMU (type "MEANSMU" at the DOS prompt.

- The program will prompt the user for the name of the input data file. Enter the full name of the input data file including the extension.

- The program will prompt the user for the name of the output file which will be created by the program. Enter the desired file name. Do not use the name of an existing file or the program will terminate.

- The program will prompt the user for the FFT smoothing window size. The window size specifies the cutoff frequency of the low pass frequency domain filter. The low pass filtered data produces an interim "local" mean or local smoothed velocity which is then ensemble averaged. Table IV-2 in Chapter IV presents some examples of the relationship between moving average window size and its associated low pass filter cutoff frequency. Other cutoff frequencies may be calculated using equation A-11 in Appendix A.

- The program will then read the input data file, compute the non-stationary mean and write the output data file. Remember to observe the column lengths of the array ENSMBL to insure that the ensemble is being constructed properly.

2. Output Description

Program MEANSMU computes the non-stationary mean and writes an output file consisting of two columns. The first column is ensemble sample number (or row number, running from 1 to LASTI). Knowing the sample rate, the time scale may be

inferred from sample number. The second column contains the non-stationary mean values. The following pages present the source code for program ENSEMBL.

```
      PROGRAM MEANSMU
*************************************************************************
*   Program to perform ensemble averaging then frequency domain low-pass
*   digital filter smoothing on an ensemble of propeller pulses.
*   The data is ensemble averaged first, then the average is filtered.
*   In other words, this program computes the time varying mean of a
*   non-stationary time series.

*   The cutoff frequency of the low pass filter is specified by the
*   size of the smoothing window, PTS, the sample rate and the number
*   of points to be smoothed, M. See the table below:
*
*         Window Size   -3 dB Cutoff Frequency (FS=30000,M=2048)
*              2                8120
*              5                3250
*             10                1625
*             15                1080
*             20                 810
*             25                 650
*             30                 540
*             35                 465
*             40                 405
*             50                 325
*
*   For other sample rates and window sizes, the cutoff frequency may
*   be determined from the equation:
*
*                FC = (0.29289 * (FS/PTS)**2)**0.5
*
*   where FC is the cutoff frequency, and FS is the sample rate.
*   Note, the input array to the smoothing subroutine does not have to be
*   an integer power of two because the subroutine zero pads as necessary
*   to the next highest integer power of two greater than the length of
*   the data vector (in this case the length of the columns of ENSMBL,
*   which is LASTI.
*
*   Input data file is assumed to be in output format of the A/D
*   conversion system, ie. a sequential ASCII file with alternating
*   values of velocity (U) and propeller pulse (PROP), starting with
*   velocity. Smoothing is accomplished with subroutine  SMOOFT
*   which performs frequency domain digital (low pass) filtering of
*   the velocity signal. Program also calculates variance for the
*   ensemble using the ensemble averaged-then-smoothed values of velocity.
*
*   Key variables are defined below:
*
*  ENSMBL = The primary data array which contains up to 50 pulses (columns)
*           each of length up to 2048 points. The pulses are synchronized in
*           time (an ensemble) such that each point in a pulse "lines up"
*           with the others.
*  U = The measured velocity as read from the input file into array ENSMBL.
*  PROP = The propeller pulse timing marker signal. Also read from the input.
*  IPLS = Number of pulses to be read from the input file.
*  LASTI = Number of points in the columns of ENSMBL. (ie. rows in ENSMBL)
*  NPLS = (J-1) = Total number of pulses (columns in ENSMBL).
```

```
*  AVG = The ensemble averaged then FFT smoothed value of U. (Output)
*  VAR = Ensemble statistical variance of U. (Output)
*  TURBIN = "Pseudo" ensemble turbulence intensity of U. (Not based on the
*           local smoothed velocity but the final mean velocity. Similar to
*           the statistical variance.) Output
*  UINF = Free stream velocity external to the boundary layer
*  PTS = Size of the smoothing window
*
*                  Programmed by: Donald K. Johnson
*                           July 1988,NPS
*
*
**********************************************************************
      DIMENSION ENSMBL(2048,50),AVG(2048),VAR(2048),TURBIN(2048)
      INTEGER I,IPLS,LASTI,J,K,L
      REAL UINF
      CHARACTER*12 INFILE,OUTFILE
C
      PRINT '(/A/)',' Ensemble Averaging and Low-Pass Filter Smoothing'
C
      PRINT '(/A\)',' ENTER THE INPUT DATA FILE NAME: '
      READ (*,'(A)') INFILE
C
C     PRINT '(/A\)',' ENTER THE NUMBER OF PROP PULSES YOU WANT: '
C     READ (*,*) IPLS
      IPLS=50
C
      PRINT '(/A\)',' ENTER THE OUTPUT DATA FILE NAME: '
      READ (*,'(A)') OUTFILE
C
***   pseudo turbulence intensity calcs currently inactive ***
*     PRINT '(/A)',' ENTER U-INFINITY (REF. VELOCITY FOR TURB.INTENSITY
*    1 CALCS) - a REAL number: '
*     READ(*,*) UINF
*     UINF=UINF*1.
      UINF=1.
C
      PRINT'(/A\)',' ENTER THE FFT SMOOTHING WINDOW SIZE: '
      READ(*,*) PTS
C
      OPEN(10,FILE=INFILE,STATUS='OLD',FORM='FORMATTED')
      OPEN(11,FILE=OUTFILE,STATUS='NEW')
C
      LASTI=2048
      J=1
      I=0
      PRINT'(/A/)',' Computing ensemble then FFT smoothed average:'
      PRINT'(/A/)',' READING INPUT FILE....'
C
 10   CONTINUE
***     Read the data into array "ENSMBL", each column is a pulse ***
        READ (10,*,END=20,ERR=20) U,PROP
        I=I+1
***     But skip the first partial pulse ******
        IF (J .GT. 1) ENSMBL(I,J-1)=U
```

```
***     If you have a new pulse, start a new column (prop trigger currently
*       set to 1. volt)
        IF(I .GE. 20 .AND. PROP .GT. 1.) THEN
            IF (J .EQ. 1) THEN
               WRITE(*,*)'Skipped first',I,' points (not a full pulse).'
            ELSE
               WRITE(*,*)'No. of points in column',J-1,' is',I
            ENDIF
***         Find the shortest column ***
            IF(I .LT. LASTI  .AND.  J .GT. 1) LASTI=I
            IF (J-1 .GE. IPLS) GO TO 28
            J=J+1
            I=0
        ENDIF
      GO TO 10
 20   CONTINUE
         WRITE(*,'(/A/)')' End of File or read error encountered - delet
     ling last partial column....'
         J=J-1
 28   CONTINUE
      XN=FLOAT(J-1)
      NPLS=J-1
C
      WRITE(*,*)'The final number of pulses is',J-1,' each of length',
     1 LASTI
*** Compute the ensemble mean, variance and "pseudo" turbulence intensity
*   based on user input value of U - infinity (free stream velocity)
*   and the smoothed mean velocity.
C
*** compute ensemble mean ***
      DO 22 I=1,LASTI
        SUM=0.
        DO 30 L=1,NPLS
           SUM=SUM+ENSMBL(I,L)
  30    CONTINUE
        AVG(I)=SUM/XN
  22  CONTINUE
*** smooth the ensemble average ***
      PRINT'(/A/)',' Smoothing ensemble average....'
      CALL SMOOFT(AVG,LASTI,PTS)
*** compute variance and pseudo turbulence inten. ***
      PRINT'(/A/)',' Computing turbulence intensity....'
      DO 50 I=1,LASTI
         SUMSQ=0.
         DO 52 L=1,NPLS
           SUMSQ=SUMSQ+(ENSMBL(I,L)-AVG(I))**2.
  52     CONTINUE
         VAR(I) = SUMSQ/(XN-1.)
         TURBIN(I) = ((SUMSQ/XN)**.5)/UINF
  50  CONTINUE
*** Write the output ******
      PRINT'(/A/)',' Outputting...'
       DO 40 I=1,LASTI
         XI=FLOAT(I)
***      note: variance and pseudo turbulence intensity is currently
```

```
*        not output ***
         WRITE(11,200) XI,AVG(I)
 200     FORMAT(F5.0,F8.1)
  40  CONTINUE
      CLOSE(10)
      CLOSE (11)
      END
C*********************************************************************
       SUBROUTINE SMOOFT(Y,N,PTS)
C---------------------------------------------------------------------
C  Y IS THE DATA ARRAY, N IS THE LENGTH OF THE DATA ARRAY, AND PTS IS
C  THE SMOOTHING WINDOW WIDTH
C
C  This subroutine is from the book "Numerical Reciepes" by Press,
C  Flannery, Teukolsky and Vetterling, Pub. 1986, Cambridge Univ. Press
C---------------------------------------------------------------------
      PARAMETER(MMAX=2048)
      DIMENSION Y(MMAX)
C
C  M IS CALCULATED TO BE THE SMALLEST POWER OF 2 WHICH IS GREATER THAN
C  THE NUMBER OF POINTS N. THE REMAINDER OF THE ARRAY IS ZERO PADDED.
C
      M=2
      NMIN=N+2.*PTS
1     IF(M.LT.NMIN)THEN
        M=2*M
      GO TO 1
      ENDIF
      IF(M.GT.MMAX) PAUSE 'MMAX too small'
      CONST=(PTS/M)**2
      Y1=Y(1)
      YN=Y(N)
      RN1=1./(N-1.)
C
C  REMOVE THE LINEAR TREND
C
      DO 11 J=1,N
        Y(J)=Y(J)-RN1*(Y1*(N-J)+YN*(J-1))
11    CONTINUE
      IF(N+1.LE.M)THEN
C
C  ZERO PAD
C
        DO 12 J=N+1,M
          Y(J)=0.
12      CONTINUE
      ENDIF
C
C TRANSFORM TO FREQUENCY DOMAIN
C
      MO2=M/2
      CALL REALFT(Y,MO2,1)
      Y(1)=Y(1)/MO2
      FAC=1.
C
```

```
C     WRITE(11,*)'     BIN (I)          Y(F)'
C     DO 99 I=1,M
C       WRITE(11,*) I,Y(I)
C 99  CONTINUE
C
C  WINDOW FUNCTION -  (USES THE PARABOLIC WELCH WINDOW)
C
C     WRITE(11,*) ' WINDOW FUNCTION:'
      DO 13 J=1,MO2-1
        K=2*J+1
        IF(FAC.NE.0.)THEN
          FAC=AMAX1(0.,(1.-CONST*J**2)/MO2)
C         WRITE(11,*) K,K+1,FAC
          Y(K)=FAC*Y(K)
          Y(K+1)=FAC*Y(K+1)
        ELSE
          Y(K)=0.
          Y(K+1)=0.
        ENDIF
13    CONTINUE
C
C  HANDLES THE LAST POINT
C
      FAC=AMAX1(0.,(1.-0.25*PTS**2)/MO2)
      Y(2)=FAC*Y(2)
C     WRITE(11,*) ' WINDOW x DATA:'
C     DO 98 I=1,M
C     WRITE(11,*) I,Y(I)
C 98  CONTINUE
C
C  TRANSFORM BACK TO TIME DOMAIN
C
      CALL REALFT(Y,MO2,-1)
C
C  RESTORE LINEAR TREND
C
      DO 14 J=1,N
        Y(J)=RN1*(Y1*(N-J)+YN*(J-1))+Y(J)
14    CONTINUE
      RETURN
      END
C*********************************************************************
      SUBROUTINE REALFT(DATA,N,ISIGN)
C---------------------------------------------------------------------
C  CALCULATES THE FOURIER TRANSFORM OF A SET OF 2*N REAL VALUED DATA
C  POINTS (ARRAY 'DATA'). ISIGN IS 1 FOR THE FOURIER TRANSFORM (TIME
C  TO FREQ. DOMAIN) AND ISIGN IS -1 FOR INVERSE TRANSFORMATIOM.
C  THIS SUBROUTINE SETS UP THE DATA TO BE EFFICIENTLY TRANSFORMED BY
C  SUBROUTINE 'FOUR1'.
C
C  This subroutine is from the book "Numerical Reciepes" by Press,
C  Flannery, Teukolsky and Vetterling, Pub. 1986, Cambridge Univ. Press
C---------------------------------------------------------------------
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION DATA(*)
```

```
      THETA=6.28318530717959D0/2.0D0/DBLE(N)
      C1=0.5
C
C  FORWARD TRANSFORM BY FOUR1
C
      IF (ISIGN.EQ.1) THEN
        C2=-0.5
        CALL FOUR1(DATA,N,+1)
      ELSE
C
C  SET UP FOR INVERSE TRANSFORM
C
        C2=0.5
        THETA=-THETA
      ENDIF
      WPR=-2.0D0*DSIN(0.5D0*THETA)**2
      WPI=DSIN(THETA)
      WR=1.0D0+WPR
      WI=WPI
      N2P3=2*N+3
      DO 11 I=2,N/2+1
        I1=2*I-1
        I2=I1+1
        I3=N2P3-I2
        I4=I3+1
        WRS=SNGL(WR)
        WIS=SNGL(WI)
        H1R=C1*(DATA(I1)+DATA(I3))
        H1I=C1*(DATA(I2)-DATA(I4))
        H2R=-C2*(DATA(I2)+DATA(I4))
        H2I=C2*(DATA(I1)-DATA(I3))
        DATA(I1)=H1R+WRS*H2R-WIS*H2I
        DATA(I2)=H1I+WRS*H2I+WIS*H2R
        DATA(I3)=H1R-WRS*H2R+WIS*H2I
        DATA(I4)=-H1I+WRS*H2I+WIS*H2R
        WTEMP=WR
        WR=WR*WPR-WI*WPI+WR
        WI=WI*WPR+WTEMP*WPI+WI
11    CONTINUE
      IF (ISIGN.EQ.1) THEN
        H1R=DATA(1)
        DATA(1)=H1R+DATA(2)
        DATA(2)=H1R-DATA(2)
      ELSE
        H1R=DATA(1)
        DATA(1)=C1*(H1R+DATA(2))
        DATA(2)=C1*(H1R-DATA(2))
C
C  INVERSE TRANSFORM
C
        CALL FOUR1(DATA,N,-1)
      ENDIF
      RETURN
      END
C*************************************************************
```

```
      SUBROUTINE FOUR1(DATA,NN,ISIGN)
C-------------------------------------------------------------
C  COMPUTES THE COMPLEX DFT OR INVERSE DFT OF ARRAY 'DATA' OF
C  LENGTH 'NN'. ISIGN IS 1 FOR THE DFT AND -1 FOR THE INVERSE
C  DFT.
C
C  This subroutine is from the book "Numerical Reciepes" by Press,
C  Flannery, Teukolsky and Vetterling, Pub. 1986, Cambridge Univ. Press
C-------------------------------------------------------------
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION DATA(*)
      N=2*NN
      J=1
      DO 11 I=1,N,2
        IF(J.GT.I)THEN
          TEMPR=DATA(J)
          TEMPI=DATA(J+1)
          DATA(J)=DATA(I)
          DATA(J+1)=DATA(I+1)
          DATA(I)=TEMPR
          DATA(I+1)=TEMPI
        ENDIF
        M=N/2
1       IF ((M.GE.2).AND.(J.GT.M)) THEN
          J=J-M
          M=M/2
        GO TO 1
        ENDIF
        J=J+M
11    CONTINUE
      MMAX=2
2     IF (N.GT.MMAX) THEN
        ISTEP=2*MMAX
        THETA=6.28318530717959D0/(ISIGN*MMAX)
        WPR=-2.D0*DSIN(0.5D0*THETA)**2
        WPI=DSIN(THETA)
        WR=1.D0
        WI=0.D0
        DO 13 M=1,MMAX,2
          DO 12 I=M,N,ISTEP
            J=I+MMAX
            TEMPR=SNGL(WR)*DATA(J)-SNGL(WI)*DATA(J+1)
            TEMPI=SNGL(WR)*DATA(J+1)+SNGL(WI)*DATA(J)
            DATA(J)=DATA(I)-TEMPR
            DATA(J+1)=DATA(I+1)-TEMPI
            DATA(I)=DATA(I)+TEMPR
            DATA(I+1)=DATA(I+1)+TEMPI
12        CONTINUE
          WTEMP=WR
          WR=WR*WPR-WI*WPI+WR
          WI=WI*WPR+WTEMP*WPI+WI
13      CONTINUE
        MMAX=ISTEP
      GO TO 2
      ENDIF

      RETURN
      END
```

F. USER'S GUIDE FOR PROGRAM TURINTMA

1. Input Instructions

- Insure that the executable file for program TURINTMA (TURINTMA.EXE) and the input data file are in the same subdirectory of a disk drive with sufficient space remaining to write the output file.

- Initiate program TURINTMA (type "TURINTMA" at the DOS prompt).

- The program will prompt the user for the name of the input data file. Enter the full name of the input data file including the extension.

- The program will prompt the user for the name of the output file which will be created by the program. Enter the desired file name. Do not use the name of an existing file or the program will terminate.

- The program will prompt the user for the moving average window size. The window size specifies the number of samples which are time averaged to produce an interim "local" mean or local smoothed velocity which is used to compute the fluctuating velocity component. The window size should be an odd number. Table IV-12 in Chapter IV presents some examples of the relationship between moving average window size and its associated low pass filter cutoff frequency. Other cutoff frequencies may be determined by solving equation A-7 in Appendix A. Program FCMA is provided to solve equation A-7.

- The program will prompt the user for the reference velocity used to normalize the turbulence intensity values. The mean velocity at the outer edge of the boundary layer (U_{edge}) is the typical reference velocity. The user will need to determine this value prior to executing this program.

- The program will then read the input data file, compute the turbulence intensity and write the output data file. Remember to observe the column lengths of the array ENSMBL to insure that the ensemble is being constructed properly.

2. Output Description

Program TURINTMA computes the non-stationary turbulence intensity, then writes an output file consisting of two columns. The first column is ensemble sample number (or row number, running from 1 to LASTI). Knowing the sample rate, the time scale may be inferred from sample number. The second column contains the non-stationary turbulence intensity values. The following pages present the source code for program ENSEMBL.

```
$DEBUG
      PROGRAM TINTMA
C*************************************************************************
*    Program computes turbulence intensity as the square root of the ensemble
*    average of the mean square fluctuating velocity component. The
*    fluctuating velocity component is the difference between the
*    instantaneous velocity and the local smoothed velocity, computed
*    using a moving average smoothing algorithm. The turbulence intensity is
*    normalized using an input reference velocity. Note that the moving average
*    operation sets the first (L-1)/2 points and the last (L-1)/2 points
*    of the mean velocity output to zero (where L is the moving average
*    window size).
*
*    The relationship between smoothing window size and the corresponding
*    cutoff frequency of the smoothing operation is given by:
*
*     SIN(((L-1)/2 + .5)*2*PI*FC/FS) / L*SIN(PI*FC/FS) = 0.7071
*
*    where FC is the cutoff frequency, FS is the sample rate and L is the
*    odd numbered window size.
*
*    Input data file is assumed to be in output format of the A/D
*    system, i.e. a sequential file with alternating values
*    of velocity (U) and propeller pulse (PROP), starting with velocity.
*    The program first separates the single time history file into an
*    ensemble of pulses using the value of the propeller trigger (PROP)
*    as the marker for the beginning of a pulse.
*
*    Key variables are defined below:
*
*   ENSMBL = The primary data array which contains up to 50 pulses (columns)
*            each of length up to 2048 points. The pulses are synchronized in
*            time (an ensemble) such that the beginning of each column (i.e.
*            the first row) represents the beginning of each pulse.
*   U = The measured velocity as read from the input file into array ENSMBL.
*   PROP = The propeller pulse timing marker signal. Also read from the input.
*   IPLS = Number of pulses to be read from the input file.
*   LASTI = Number of points in the columns of ENSMBL. (ie. rows in ENSMBL)
*   NPLS = (J-1) = Final total number of pulses (columns in ENSMBL).
*   L = The width of the moving average window  (should be an odd number).
*   A(I) = The current column of ENSMBL to be smoothed using subroutine MOVAVE
*   B(I) = The current smoothed column of ENSMBL. (Output of MOVAVE)
*   TURBIN(I) = The average turbulence intensity for NPLS propeller pulses.
*   UEDGE = The reference velocity used to normalize the turbulence intensity
*           (the mean velocity at the outer edge of the boundary layer is
*           usually used - input from keyboard).
*
*                    Programmed by: Donald K. Johnson
*                           AUGUST,1988,NPS
*
*************************************************************************
      PARAMETER (MMAX=2048)
      DIMENSION ENSMBL(2048,50),A(MMAX),B(MMAX),TURBIN(MMAX)
      INTEGER L,NPTS,ILAST,NPLS,IPLS
      REAL UEDGE
```

```
      CHARACTER*12 INFILE,OUTFILE
*
      PRINT'(/A/)',' Turbulence Intensity from Moving Average Smoothing'
*
      PRINT '(/A\)',' ENTER THE INPUT DATA FILE NAME: '
      READ (*,'(A)') INFILE
*
      PRINT '(/A\)',' ENTER THE OUTPUT FILE NAME: '
      READ (*,'(A)') OUTFILE
*
      PRINT '(/A\)',' ENTER THE MOVING AVERAGE WINDOW SIZE" (AN ODD NUMB
     1ER): '
      READ (*,*) L
*
      PRINT '(/A\)',' ENTER U-EDGE (REFERENCE VELOCITY FOR TURBULENCE IN
     1TENSITY NORMALIZATION): '
      READ(*,*) UEDGE
      UEDGE=UEDGE*1.
*
      OPEN(UNIT=10,FILE=INFILE,STATUS='OLD')
      OPEN(UNIT=11,FILE=OUTFILE,STATUS='NEW')
*
***   separate the time history data into an ensemble of pulses ***
      LASTI=2048
      IPLS=50
      J=1
      I=0
      PRINT'(/A/)',' READING INPUT FILE....'
      PRINT'(/A/)',' WARNING, Watch the number of points in each column
     1... they should all be about the same length.'
*
 10   CONTINUE
***     Read the data into array "ENSMBL", each column is a pulse ***
        READ (10,*,END=20,ERR=20) U,PROP
        I=I+1
***     But skip the first partial pulse ******
        IF (J .GT. 1) ENSMBL(I,J-1)=U
***     If you have a new pulse, start a new column. ***
*       Note, prop trigger is set to 1. volt to start a new pulse ***
        IF(I .GE. 20 .AND. PROP .GT. 1.) THEN
            IF (J .EQ. 1) THEN
               WRITE(*,*)'Skipped first',I,' points (not a full pulse).'
            ELSE
               WRITE(*,*)'No. of points in column',J-1,' is',I
            ENDIF
***         Find the shortest column ***
            IF(I .LT. LASTI  .AND.  J .GT. 1) LASTI=I
***         Quit loop if you have enough pulses ***
            IF (J-1 .GE. IPLS) GO TO 28
            J=J+1
            I=0
        ENDIF
      GO TO 10
 20   CONTINUE
        WRITE(*,'(/A/)')' End of File encountered - deleting last parti
```

```
     1al column....'
         J=J-1
 28   CONTINUE
      XN=FLOAT(J-1)
      NPLS=J-1
*
      WRITE(*,*)'The final number of pulses is',NPLS,' each of length',
     1 LASTI
*
      PRINT '(/A/)',' Working...'
*
***   apply the moving average window to each column (pulse) in array ENSMBL
*
***   start the pulse loop ***
      DO 25 K=1,NPLS
***      and start the time loop ***
         DO 15 I=1,LASTI
***         copy the current (Kth) pulse into A for call to subroutine ***
            A(I)=ENSMBL(I,K)
 15      CONTINUE
***      smooth the Kth pulse ***
         CALL MOVAVE(A,LASTI,L,B)
*
***      compute the square of the fluctuating velocity component
*        and write value back into ENSMBL
         DO 16 I=1,LASTI
            ENSMBL(I,K)=(ENSMBL(I,K)-B(I))**2.
 16      CONTINUE
 25   CONTINUE
*
***   now ensemble average (across rows) to get the mean square fluctuating
*     velocity.
***   start with a time loop ***
      DO 50 I=1,LASTI
***   now sum across each row ***
      SUMU=0.
         DO 60 K=1,NPLS
           SUMU=SUMU+ENSMBL(I,K)
 60      CONTINUE
*** compute the normalized turbulence intensity (the root mean square) ***
         TURBIN(I)=(SUMU/NPLS)**.5/UEDGE
         IF(I .LE. (L-1)/2  .OR.  I .GT. LASTI-(L-1)/2 ) TURBIN(I)=0.
 50   CONTINUE
***   write the output ***
      DO 70 I=1,LASTI
         XI=FLOAT(I)
         WRITE(11,100) XI,TURBIN(I)
 100     FORMAT(1X,F5.0,1X,F10.5)
 70   CONTINUE
      CLOSE(10)
      CLOSE(11)
      END
C*********************************************************************
      SUBROUTINE MOVAVE(A,N,L,B)
*
```

```
*    Program performs a simple moving average of a time series to determine a
*    local mean (smoothed) velocity. Note that the first L/2 values and the
*    last L/2 values of the input time series are set to zero. Briefly, the
*    moving average accomplishes a crude low pass filter type of smoothing
*    on the input time series as follows. Starting with the first L (usually
*    odd) points in the time series, a time average is computed for the L
*    points. This average value becomes the smoothed velocity at the center
*    of the L points, point (L-1)/2 for odd L. Then the moving average window
*    is moved over one point and a new average is computed for the window. This
*    is repeated until the Lth point of the window reaches the end of the time
*    series.
*
*    The key variables are defined below:
*
*    A = A column array containing the time series to be averaged.
*    N = The number of values in A.
*    L = The size of the averaging window (an odd number is best because
*        then the average of the small time segment is centered on the
*        mid-point of the segment). Note, a larger window size corresponds to
*        a lower low pass filter cutoff frequency. See the table below for the
*        relation between window size and cutoff frequency.
*    B = A column array containing the smoothed (averaged) time series values.
*
      DIMENSION A(1),B(1)
      AL=L
      MH=L/2
***   zero out the answer array ***
      DO 18 I=1,N
   18    B(I)=0.
      K=L-1
***   if L is .GT. N then quit ***
      IF (N-L) 1,2,2
    2 NN=N-K
***   start the main averaging loop, working on L/2 points on either
*     side of the KK th data point. (KK is calculated below)
      DO 4 J=1,NN
         JN=J+K
***      reset the accumulator ***
         S=0.
***      sum across L points centered on point KK.
         DO 3 I=J,JN
            S=S+A(I)
    3    CONTINUE
***      now divide by L to get the average of segment centered on point KK.
         KK=J+MH
         B(KK)=S/AL
    4 CONTINUE
***   if L is even then take the average of two adjacent terms to get the value
*     of the average velocity centered on the KK th point.
      IF (L-(L/2)*2) 1,6,1
    6 MS=MH+1
      MD=MH+NN-1
      DO 7 I=MS,MD
         TEMP=(B(I)+B(I+1))/2.
         B(I)=TEMP

    7 CONTINUE
    1 RETURN
      END
```

G. USER'S GUIDE FOR PROGRAM TURBIN

1. Input Instructions

- Insure that the executable file for program TURBIN (TURBIN.EXE) and the input data file are in the same subdirectory of a disk drive with sufficient space remaining to write the output file.

- Initiate program TURBIN (type "TURBIN" at the DOS prompt).

- The program will prompt the user for the name of the input data file. Enter the full name of the input data file including the extension.

- The program will prompt the user for the name of the output file which will be created by the program. Enter the desired file name. Do not use the name of an existing file or the program will terminate.

- The program will prompt the user for the FFT smoothing window size. The window size specifies the cutoff frequency of the low pass frequency domain filter. The low pass filtered data produces an interim "local" mean or local smoothed velocity which is used to compute the fluctuating velocity component. Table IV-23 in Chapter IV presents some examples of the relationship between moving average window size and its associated low pass filter cutoff frequency. Other cutoff frequencies may be calculated using equation A-11 in Appendix A.

- The program will prompt the user for the reference velocity used to normalize the turbulence intensity values. The mean velocity at the outer edge of the boundary layer (U_{edge}) is the typical reference velocity. The user will need to determine this value prior to executing this program.

- The program will then read the input data file, compute the turbulence intensity and write the output data file. Remember to observe the column lengths of the array ENSMBL to insure that the ensemble is being constructed properly.

2. Output Description

Program TURBIN computes the non-stationary turbulence intensity, then writes an output file consisting of two columns. The first column is ensemble sample number (or row number, running from 1 to LASTI). Knowing the sample rate, the time scale may be inferred from sample number. The second column contains the non-stationary turbulence intensity values. The following pages present the source code for program ENSEMBL.

```
      PROGRAM TURINT
*************************************************************************
*   Program computes the non-stationary turbulence intensity of an ensemble
*   of propeller pulses by calculating the RMS of the fluctuating velocity.
*   The mean in this case is an ensemble average. The fluctuating component
*   is the difference between the instantaneous velocity and the local
*   smoothed velocity, computed using a frequency domain low pass filter
*   smoothing  routine. The turbulence intensity is normalized using an
*   input reference velocity. This program is the turbulence intensity
*   counterpart to the non-stationary mean program - MEANSMU.
*
*   The cutoff frequency of the low pass smoothing filter is specified by the
*   size of the smoothing window, PTS, the sample rate and the number
*   of points to be smoothed, M. See the table below:
*
*         Window Size    -3 dB Cutoff Frequency (FS=30000,M=2048)
*              2                 8120
*              5                 3250
*             10                 1625
*             15                 1080
*             20                  810
*             25                  650
*             30                  540
*             35                  465
*             40                  405
*             50                  325
*
*   For other sample rates and window sizes, the cutoff frequency may
*   be determined from the equation:
*
*                FC = (0.29289 * (FS/PTS)**2)**0.5
*
*   where FC is the cutoff frequency, and FS is the sample rate.
*   Note, the input array to the smoothing subroutine does not have to be
*   an integer power of two because the subroutine zero pads as necessary
*   to the next highest integer power of two greater than the length of
*   the data vector (in this case the length of the columns of ENSMBL,
*   which is LASTI.
*
*   Input data file is assumed to be in output format of the Labtech
*   Notebook system supporting the D/A board, ie. a sequential
*   file with alternating values of velocity (U) and propellor pulse
*   (PROP), starting with velocity.
*
*   Key variables are defined below:
*
*  ENSMBL = The primary data array which contains up to 50 pulses (columns)
*           each of length up to 2048 points. The pulses are synchronized in
*           time (an ensemble) such that each point in a pulse "lines up"
*           with the corresponding points in other pulses.
*  U = The measured velocity as read from the input file into array ENSMBL.
*  PROP = The propeller pulse timing marker signal. Also read from the input.
*  IPLS = Number of pulses to be read from the input file.
*  LASTI = Number of points in the columns of ENSMBL. (ie. rows in ENSMBL)
*  NPLS = (J-1) = Total number of pulses (columns in ENSMBL).
```

```
*  VAR = Ensemble variance of U.
*  TURBIN = Ensemble turbulence intensity of U.
*  UINF = Free stream velocity external to the boundary layer
*
*                  Programmed by: Donald K. Johnson
*                           July 1988,NPS
*
*
*************************************************************************
      DIMENSION ENSMBL(2048,50),VAR(2048),TURBIN(2048)
      DIMENSION UCOL(2048)
      INTEGER I,IPLS,LASTI,J,K,L
      CHARACTER*12 INFILE,OUTFILE
C
      PRINT '(/A/)',' Turbulence Intensity from Frequency Domain Low Pas
     1s Filter Smoothing '
C
      PRINT '(/A\)',' ENTER THE INPUT DATA FILE NAME: '
      READ (*,'(A)') INFILE
C
*      PRINT '(/A\)',' ENTER THE NUMBER OF PROP PULSES YOU WANT: '
*      READ (*,*) IPLS
       IPLS=50
C
      PRINT '(/A\)',' ENTER THE OUTPUT DATA FILE NAME: '
      READ (*,'(A)') OUTFILE
C
      PRINT '(/A)',' ENTER U-EDGE (REFERENCE VELOCITY FOR TURBULENCE INT
     1ENSITY NORMALIZATION): '
      READ(*,*) UINF
      UINF=UINF*1.
C
      PRINT'(/A\)',' ENTER THE FFT SMOOTHING WINDOW SIZE: '
      READ(*,*) PTS
C
      OPEN(10,FILE=INFILE,STATUS='OLD',FORM='FORMATTED')
      OPEN(11,FILE=OUTFILE,STATUS='NEW')
C
      LASTI=2048
      J=1
      I=0
      PRINT'(/A/)',' Computes ensemble average of turb. inten based on
     1local (smoothed) mean:'
      PRINT '(/A/)',' READING INPUT FILE....'
      PRINT '(A/)',' CAUTION, Watch the number of points in each column
     1 ... they should all be about the same length.'
C
 10   CONTINUE
***     Read the data into array "ENSMBL", each column is a pulse ***
        READ (10,*,END=20,ERR=20) U,PROP
        I=I+1
***     But skip the first partial pulse ******
        IF (J .GT. 1) ENSMBL(I,J-1)=U
***     If you have a new pulse, start a new column (prop trigger currently
*       set to 1. volt). ***
```

```
        IF(I .GE. 20 .AND. PROP .GT. 1.) THEN
            IF (J .EQ. 1) THEN
               WRITE(*,*)'Skipped first',I,' points (not a full pulse).'
            ELSE
               WRITE(*,*)'No. of points in column',J-1,' is',I
            ENDIF
***         Find the shortest column ***
            IF(I .LT. LASTI  .AND.  J .GT. 1) LASTI=I
            IF (J-1 .GE. IPLS) GO TO 28
            J=J+1
            I=0
         ENDIF
      GO TO 10
 20   CONTINUE
         PRINT '(/A/)',' End of File encountered - deleting last partial
     1 column....'
         J=J-1
 28   CONTINUE
      XN=FLOAT(J-1)
      NPLS=J-1
C
      WRITE(*,*) 'The final number of pulses is',NPLS,' each of length',
     1LASTI
C
*** Perform the FFT smoothing on each column of the array ENSMBL ***
      DO 300 L=1,NPLS
      WRITE(*,*)'Smoothing column',L
         DO 310 I=1,LASTI
            UCOL(I)=ENSMBL(I,L)
 310     CONTINUE
         CALL SMOOFT(UCOL,LASTI,PTS)
***      Write the square error term [(U-Usmooth)**2] back into ENSMBL ***
         DO 320 I=1,LASTI
            ENSMBL(I,L)=(ENSMBL(I,L)-UCOL(I))**2
 320     CONTINUE
 300  CONTINUE
C
*** Compute the average local variance and turbulence intensity based on
*   user input value of U - edge (reference velocity), and based on
*   the variance of U from the local smoothed U. Note, the variance is not
*   the true statistical variance.
      DO 22 I=1,LASTI
        SUM=0.
        SUMSQ=0.
        DO 30 L=1,J-1
           SUMSQ=SUMSQ+ENSMBL(I,L)
  30    CONTINUE
        VAR(I)=(SUMSQ)/(XN-1.)
        TURBIN(I)=((SUMSQ/XN)**.5)/UINF
  22  CONTINUE
*** Write the output ******
       DO 40 I=1,LASTI
         XI=FLOAT(I)
***      note: variance is currently not output, write array VAR if desired
         WRITE(11,200) XI,TURBIN(I)
```

```
 200     FORMAT(1X,F5.0,F10.5)
  40  CONTINUE
      CLOSE(10)
      CLOSE (11)
      END
C*********************************************************************
      SUBROUTINE SMOOFT(Y,N,PTS)
C---------------------------------------------------------------------
C  Y IS THE DATA ARRAY, N IS THE LENGTH OF THE DATA ARRAY, AND PTS IS
C  THE SMOOTHING WINDOW WIDTH
C
C  This subroutine is from the book "Numerical Reciepes" by Press,
C  Flannery, Teukolsky and Vetterling, Pub. 1986, Cambridge Univ. Press
C---------------------------------------------------------------------
      PARAMETER(MMAX=2048)
      DIMENSION Y(MMAX)
C
C  M IS CALCULATED TO BE THE SMALLEST POWER OF 2 WHICH IS GREATER THAN
C  THE NUMBER OF POINTS N. THE REMAINDER OF THE ARRAY IS ZERO PADDED.
C
      M=2
      NMIN=N+2.*PTS
1     IF(M.LT.NMIN)THEN
        M=2*M
      GO TO 1
      ENDIF
      IF(M.GT.MMAX) PAUSE 'MMAX too small'
      CONST=(PTS/M)**2
      Y1=Y(1)
      YN=Y(N)
      RN1=1./(N-1.)
C
C  REMOVE THE LINEAR TREND
C
      DO 11 J=1,N
        Y(J)=Y(J)-RN1*(Y1*(N-J)+YN*(J-1))
11    CONTINUE
      IF(N+1.LE.M)THEN
C
C  ZERO PAD
C
        DO 12 J=N+1,M
          Y(J)=0.
12      CONTINUE
      ENDIF
C
C TRANSFORM TO FREQUENCY DOMAIN
C
      MO2=M/2
      CALL REALFT(Y,MO2,1)
      Y(1)=Y(1)/MO2
      FAC=1.
C
C     WRITE(11,*)'     BIN (I)          Y(F)'
C     DO 99 I=1,M
```

```
C       WRITE(11,*) I,Y(I)
C 99  CONTINUE
C
C  WINDOW FUNCTION -  (USES THE PARABOLIC WELCH WINDOW)
C
C     WRITE(11,*) ' WINDOW FUNCTION:'
      DO 13 J=1,MO2-1
        K=2*J+1
        IF(FAC.NE.0.)THEN
          FAC=AMAX1(0.,(1.-CONST*J**2)/MO2)
C         WRITE(11,*) K,K+1,FAC
          Y(K)=FAC*Y(K)
          Y(K+1)=FAC*Y(K+1)
        ELSE
          Y(K)=0.
          Y(K+1)=0.
        ENDIF
13    CONTINUE
C
C  HANDLES THE LAST POINT
C
      FAC=AMAX1(0.,(1.-0.25*PTS**2)/MO2)
      Y(2)=FAC*Y(2)
C     WRITE(11,*) ' WINDOW x DATA:'
C     DO 98 I=1,M
C     WRITE(11,*) I,Y(I)
C 98  CONTINUE
C
C  TRANSFORM BACK TO TIME DOMAIN
C
      CALL REALFT(Y,MO2,-1)
C
C  RESTORE LINEAR TREND
C
      DO 14 J=1,N
        Y(J)=RN1*(Y1*(N-J)+YN*(J-1))+Y(J)
14    CONTINUE
      RETURN
      END
C*********************************************************************
      SUBROUTINE REALFT(DATA,N,ISIGN)
C---------------------------------------------------------------------
C  CALCULATES THE FOURIER TRANSFORM OF A SET OF 2*N REAL VALUED DATA
C  POINTS (ARRAY 'DATA'). ISIGN IS 1 FOR THE FOURIER TRANSFORM (TIME
C  TO FREQ. DOMAIN) AND ISIGN IS -1 FOR INVERSE TRANSFORMATIOM.
C  THIS SUBROUTINE SETS UP THE DATA TO BE EFFICIENTLY TRANSFORMED BY
C  SUBROUTINE 'FOUR1'.
C
C  This subroutine is from the book "Numerical Reciepes" by Press,
C  Flannery, Teukolsky and Vetterling, Pub. 1986, Cambridge Univ. Press
C---------------------------------------------------------------------
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION DATA(*)
      THETA=6.28318530717959D0/2.0D0/DBLE(N)
      C1=0.5
```

```
C
C  FORWARD TRANSFORM BY FOUR1
C
      IF (ISIGN.EQ.1) THEN
        C2=-0.5
        CALL FOUR1(DATA,N,+1)
      ELSE
C
C  SET UP FOR INVERSE TRANSFORM
C
        C2=0.5
        THETA=-THETA
      ENDIF
      WPR=-2.0D0*DSIN(0.5D0*THETA)**2
      WPI=DSIN(THETA)
      WR=1.0D0+WPR
      WI=WPI
      N2P3=2*N+3
      DO 11 I=2,N/2+1
        I1=2*I-1
        I2=I1+1
        I3=N2P3-I2
        I4=I3+1
        WRS=SNGL(WR)
        WIS=SNGL(WI)
        H1R=C1*(DATA(I1)+DATA(I3))
        H1I=C1*(DATA(I2)-DATA(I4))
        H2R=-C2*(DATA(I2)+DATA(I4))
        H2I=C2*(DATA(I1)-DATA(I3))
        DATA(I1)=H1R+WRS*H2R-WIS*H2I
        DATA(I2)=H1I+WRS*H2I+WIS*H2R
        DATA(I3)=H1R-WRS*H2R+WIS*H2I
        DATA(I4)=-H1I+WRS*H2I+WIS*H2R
        WTEMP=WR
        WR=WR*WPR-WI*WPI+WR
        WI=WI*WPR+WTEMP*WPI+WI
11    CONTINUE
      IF (ISIGN.EQ.1) THEN
        H1R=DATA(1)
        DATA(1)=H1R+DATA(2)
        DATA(2)=H1R-DATA(2)
      ELSE
        H1R=DATA(1)
        DATA(1)=C1*(H1R+DATA(2))
        DATA(2)=C1*(H1R-DATA(2))
C
C  INVERSE TRANSFORM
C
        CALL FOUR1(DATA,N,-1)
      ENDIF
      RETURN
      END
C*************************************************************
      SUBROUTINE FOUR1(DATA,NN,ISIGN)
C-------------------------------------------------------------
```

```
C  COMPUTES THE COMPLEX DFT OR INVERSE DFT OF ARRAY 'DATA' OF
C  LENGTH 'NN'. ISIGN IS 1 FOR THE DFT AND -1 FOR THE INVERSE
C  DFT.
C
C  This subroutine is from the book "Numerical Reciepes" by Press,
C  Flannery, Teukolsky and Vetterling, Pub. 1986, Cambridge Univ. Press
C-----------------------------------------------------------------
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION DATA(*)
      N=2*NN
      J=1
      DO 11 I=1,N,2
        IF(J.GT.I)THEN
          TEMPR=DATA(J)
          TEMPI=DATA(J+1)
          DATA(J)=DATA(I)
          DATA(J+1)=DATA(I+1)
          DATA(I)=TEMPR
          DATA(I+1)=TEMPI
        ENDIF
        M=N/2
1       IF ((M.GE.2).AND.(J.GT.M)) THEN
          J=J-M
          M=M/2
        GO TO 1
        ENDIF
        J=J+M
11    CONTINUE
      MMAX=2
2     IF (N.GT.MMAX) THEN
        ISTEP=2*MMAX
        THETA=6.28318530717959D0/(ISIGN*MMAX)
        WPR=-2.D0*DSIN(0.5D0*THETA)**2
        WPI=DSIN(THETA)
        WR=1.D0
        WI=0.D0
        DO 13 M=1,MMAX,2
          DO 12 I=M,N,ISTEP
            J=I+MMAX
            TEMPR=SNGL(WR)*DATA(J)-SNGL(WI)*DATA(J+1)
            TEMPI=SNGL(WR)*DATA(J+1)+SNGL(WI)*DATA(J)
            DATA(J)=DATA(I)-TEMPR
            DATA(J+1)=DATA(I+1)-TEMPI
            DATA(I)=DATA(I)+TEMPR
            DATA(I+1)=DATA(I+1)+TEMPI
12        CONTINUE
          WTEMP=WR
          WR=WR*WPR-WI*WPI+WR
          WI=WI*WPR+WTEMP*WPI+WI
13      CONTINUE
        MMAX=ISTEP
      GO TO 2
      ENDIF
      RETURN
      END
```

H. USER'S GUIDE FOR PROGRAM PSD

1. Input Instructions

- Insure that the executable file for program PSD (PSD.EXE) and the input data file are in the same subdirectory of a disk drive with sufficient space remaining to write the output file. Note that even though program CORREX does not work with an ensemble, the program is currently set up to read a data file with two inputs such as U and PROP described in section B.1. The PROP variable is not used and is only a "place holder" in the read statement. The read statement is easily modified to suit the user's needs.

- Initiate program PSD (type "PSD" at the DOS prompt).

- The program will prompt the user for the name of the input data file. Enter the full name of the input data file including the extension.

- The program will prompt the user for the name of the output file which will be created by the program. Enter the desired file name. Do not use the name of an existing file or the program will terminate.

- The program will prompt the user for the sample rate of the date. Enter the sample rate of the data contained in the input file.

- The program will prompt the user to select one of the seven tapering window options from Table IV-4. Enter the number of the desired option. (Windows 5 and 6 are good compromises.)

- The program will prompt the user to choose whether or not to remove the stationary mean prior to computing the spectral estimate. Enter "Y" for yes (remove the mean) or "N" for no (do not remove the mean). If the data has a large mean value (relative to the magnitude of the oscillations) then it is recommended that the mean be removed.

- The program will prompt the user to enter the desired number of frequency bins (M) and half of the number of overlapping segments (K). Enter the selected values of M and K separated by a comma. WARNING: The values for M and K may NOT be selected arbitrarily! The allowable

values of M and K are dependent on each other and on the total number of data points in the time series (N) according to the equation N = (2K + 1)M where M must be an integer power of two and N is equal to or slightly less than an integer power of two. The tradeoff is too obtain an adequate frequency resolution (large M) and still minimize the variance of the spectral estimate sufficiently (large K). Table IV-3 presents some typical values of M and K for different record lengths N. Usually, the total number of data points (N) will not be an integer power of two. The user may choose to read a lesser number of data points (reasonable if N is large, say greater than 4096). Or the user can add zeros to the end of the data file to achieve the next larger integer power of two value for N. Note that the actual number of points read by the program is determined by the user input values of M and K.

- The program will then compute the spectral estimate and write the output file.

2. Output Description

The output file consists of five columns. The first column is the frequency. The second column contains the magnitude of the power at each listed frequency. The third column presents the relative power in decibels (dB) computed such that the peak power is zero dB. The fourth column presents a running total of the power (magnitude) and the fifth column lists the percent running total power. The following pages present the source code for program ENSEMBL.

```
      PROGRAM PSD
*************************************************************************
C  This program computes an estimate of the PSD of a real discrete time
C  series using the classical averaged periodogram method with 50% overlapped
C  segments for (nearly) minimum variance. The main program is a driver to
C  set up the input for the call to SUBROUTINE SPCTRM which does most of
C  the work including reading the data.
C
C  Input file is a sequential file containing values  of velocity and PROP.
C  (Such as the output of the Labtech Notebook A/D conversion software. Time
C  is implied by the sample rate.
C
C  The key variables are defined below:
C
C  M = Number of freq. bins and half the number of points per segment.
C      Large M implies high resolution in frequency. (MUST BE A POWER OF 2)
C  K = 1/2 of the number of segments. Large K implies low PSD variance.
C      K is given by:    K = INT((N/M - 1)/2)
C  N = Total number of data points (power of 2). N = (2K+1)M
C
C  TYPICAL EXAMPLES:           N          M           K
C
C                              512         32          7
C                                          64          3
C                                         128          1
C
C                             1024         32         15
C                                          64          7
C                                         128          3
C
C                             2048         64         15
C                                         128          7
C                                         256          3
C
C                             4096         64         31
C                                         128         15
C                                         256          7
C
C                             8192         64         63
C                                         128         31
C                                         256         15
C                                         512          7
C
C  SMPLRT = Sample rate of the input data.
C  FREQ = Frequency value in HZ of a particular frequenct bin.
C  DFREQ = The width of each frequency bin.
C  W1 and W2 = Workspace arrays for reading the input data sequence from
C              unit 9.
C  P = Output PSD array in (ft/sec)**2 per Hz
C  DBP = Output PSD array in decibels per Hz
C  T1 and T2 = Dummy read variables.
C  WINDOW = The current FFT window function value (see function subroutine).
C  WOPT = Window selection parameter:    WOPT            WINDOW
C                                         1              Parzen (triangle)
C                                         2              rectangular (none)
```

```
C                                       3               Welch
C                                       4               Van Hann
C                                       5               Hamming
C                                       6               3 term B-H
C                                       7               4 term B-H
C
C  Note on dimensions: P and PSD are currently dimensioned to ((2**L) + 1)
C  frequency bins where L=9. (That's 513, so the MAXIMUM frequency
C  resolution is currently 513 bins.) W2 and W1 are dimensioned to
C  (2**L) and 4*(2**L) respectively. (That's 512 and 2048.) If you need
C  more resolution, the next higher dimensions are 1025 for P and DBP, and
C  1024 and 4096 for W2 and W1 respectively.
C
C               Programmed by: Donald K. Johnson
C                              August 1988, NPS Monterey, CA
C*************************************************************************
      REAL SMPLRT, DFREQ, FREQ, T2, T1
      INTEGER I,J,K,M,WOPT
      DIMENSION P(513),W1(2048),W2(512),DBP(513),TPWR(513),PCTPWR(513)
      LOGICAL OVRLAP
      CHARACTER*12 INFILE,OUTFILE
      CHARACTER*1 RM
C
      PRINT '(/A/)',' Welch Periodogram Spectral Estimation Program'
C
      PRINT '(/A\)',' ENTER THE INPUT DATA FILE NAME: '
      READ (*,'(A)') INFILE
C
      PRINT '(/A\)',' ENTER THE OUTPUT FILE NAME: '
      READ (*,'(A)') OUTFILE
C
      PRINT '(/A\)',' ENTER THE SAMPLE RATE OF THE DATA: '
      READ (*,*) SMPLRT
C
      WRITE(*,'(/A\)')' ENTER FFT "WINDOW" OPTION (1 thru 7): '
      READ(*,*) WOPT
C
      PRINT '(/A\)',' DO YOU WANT TO REMOVE THE MEAN FIRST (Y OR N)? '
      READ (*,'(A)') RM
C
      PRINT '(/A\)',' ENTER THE NO. OF FREQ BINS (M) AND HALF OF THE NO.
     1 OF SEGMENTS (K): '
      READ (*,*) M,K
      M4=4*M
C
      OPEN(UNIT=9,FILE=INFILE,STATUS='OLD')
      OPEN(UNIT=11,FILE=OUTFILE,STATUS='NEW')
C
*** if selected, remove the mean and write the u' into scratch file unit 10
      KUNIT=9
      IF (RM .EQ. 'Y'  .OR.  RM .EQ. 'y') THEN
           OPEN(UNIT=10,STATUS='SCRATCH')
           NPTS=(2*K+1)*M
           PRINT '(/A\,I5)',' NPTS = ',NPTS
           SUMD1=0.
```

```
***     compute the sum
        DO 2 I=1,NPTS
*          READ(9,*) U
           READ(9,*) U,PROP
           SUMD1=SUMD1+U
 2      CONTINUE
        REWIND(9)
***     compute the mean
        AVG1=SUMD1/NPTS
        PRINT '(/A\,F12.4)',' The removed mean is: ',AVG1
***     compute u' (remove the mean)
        DO 4 I=1,NPTS
*           READ(9,*) U
            READ(9,*) U,PROP
           UPRM=U-AVG1
           WRITE(10,*) UPRM,PROP
 4      CONTINUE
        REWIND(9)
        REWIND(10)
***     change the read file unit number from 9 to 10 (passed to subroutine)
        KUNIT=10
      ENDIF
*
      DFREQ=SMPLRT/(2*M)
      OVRLAP=.TRUE.
*
*     The subroutine itself contains the actual read statements
*
      PRINT '(/A/)',' Estimating spectrum ...'
      CALL SPCTRM(P,M,K,OVRLAP,WOPT,KUNIT,W1,W2,T1,T2,DBP,TPWR,PCTPWR)
      WRITE(*,*) 'SPECTRUM OF ',INFILE,' IS IN FILE: ',OUTFILE
      DO 11 J=1,M+1
         FREQ=J*DFREQ - DFREQ
         WRITE(11,'(1X,F10.1,F17.3,F10.3,F17.3,F8.2)') FREQ,P(J),DBP(J),
     1 TPWR(J),PCTPWR(J)
  11  CONTINUE
      CLOSE(9)
      CLOSE(10)
      CLOSE(11)
      PRINT '(/,/A)','                ALL DONE!'
      END
*************************************************************************
      SUBROUTINE SPCTRM(P,M,K,OVRLAP,WOPT,KUNIT,W1,W2,T1,T2,DBP,TPWR,
     +PCTPWR)
C
C  This subroutine represents a modified version of a spectral analysis
C  program published in "Numerical Recipes" by Press, Flannery, Teukolsky,
C  and Vetterling, pub 1986, Cambridge Univ. Press.
C
C  Reads data from unit 9 and returns as P(J) the data's power (mean square
C  amplitude) at frequency (J-1)/(2*M) which is then converted to HZ in the
C  main program, for J=1 to M based on (2*K+1)*M data points.
C
C
C                       modified by: Donald K. Johnson
```

```
C                                    August 1988, NPS Monterey, CA
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      LOGICAL OVRLAP
      INTEGER WOPT
      DIMENSION P(M+1),W1(4*M),W2(M),DBP(M+1),TPWR(M+1),PCTPWR(M+1)
      MM=M+M
      M4=MM+MM
      M44=M4+4
      M43=M4+3
      DEN=0.
      FACM=M-0.5
      FACP=1./(M+0.5)
      PI2=8.*ATAN(1.)
      SUMW=0.
***   compute the sum of the squared window function for later normalizing ***
*     the window functions are given in the function subroutine below.
      DO 11 J=1,MM
        SUMW=SUMW+WINDOW(J,MM,WOPT)**2
11    CONTINUE
C *** initialize the spectrum to zero ***
      DO 12 J=1,M
        P(J)=0.
12    CONTINUE
C *** initialize the "save" half buffer ***
      IF(OVRLAP)THEN
*
*               for overlapping, W2 is the velocity and, T2 is a dummy
        READ (KUNIT,*) (W2(J),T2,J=1,M)
**        READ (KUNIT,*) (W2(J),J=1,M)
*
      ENDIF
C *** loop over the data segments in groups of two. Get two complete
C     segments into the workspace. ***
      DO 18 KK=1,K
        DO 15 JOFF=-1,0,1
          IF (OVRLAP) THEN
            DO 13 J=1,M
              W1(JOFF+J+J)=W2(J)
13          CONTINUE
*
*                 for overlapping, W2 is the velocity and T2 is a dummy
            READ (KUNIT,*) (W2(J),T2,J=1,M)
**            READ (KUNIT,*) (W2(J),J=1,M)
*
            JOFFN=JOFF+MM
            DO 14 J=1,M
              W1(JOFFN+J+J)=W2(J)
14          CONTINUE
          ELSE
*                       for no overlap, W1 is the velocity and T1 is a dummy
            READ (KUNIT,*) (W1(J),T1,J=JOFF+2,M4,2)
**            READ (KUNIT,*) (W1(J),J=JOFF+2,M4,2)
*
          ENDIF
15      CONTINUE
```

```
C *** apply the window function defined above to the data ***
        DO 16 J=1,MM
          J2=J+J
          W=WINDOW(J,MM,WOPT)
          W1(J2)=W1(J2)*W
          W1(J2-1)=W1(J2-1)*W
16      CONTINUE
C *** Fourier transform the windowed data, then sum results into previous
C     segment ***
        CALL FOUR1(W1,MM,1)
        P(1)=P(1)+W1(1)**2+W1(2)**2
        DO 17 J=2,M
          J2=J+J
          P(J)=P(J)+W1(J2)**2+W1(J2-1)**2
     *         +W1(M44-J2)**2+W1(M43-J2)**2
17      CONTINUE
        P(M+1)=P(M+1)+W1(MM+1)**2+W1(MM+2)**2
        DEN=DEN+SUMW
18    CONTINUE
C *** correct normalization ***
      DEN=M4*DEN
C *** normalize the output ***
      DO 19 J=1,M+1
        P(J)=P(J)/DEN
19    CONTINUE
C*** compute the relative PSD in decibels and the running total power ***
      PMAX=P(1)
      DO 50 J=1,M
         IF(P(J+1) .GT. PMAX) PMAX=P(J+1)
   50 CONTINUE
      SUMPWR=0.
      DO 60 J=1,M+1
         IF (P(J) .EQ. 0.) THEN
            WRITE(*,*) 'WARNING: P(J) has a zero value. P(J) set to -999
     1999.0  at a J of',J,'                          ... continuing.'
            DBP(J)=-999999.
         ELSE
            DBP(J)=20.*ALOG10(P(J)/PMAX)
            SUMPWR=SUMPWR+P(J)
            TPWR(J)=SUMPWR
         ENDIF
   60 CONTINUE
*** compute the percentage of total power at freq. J relative to the
*   total power summed over all freqs.
      DO 70 J=1,M+1
         PCTPWR(J)=100.*TPWR(J)/TPWR(M+1)
  70  CONTINUE
      RETURN
      END
C*******************************************************************************
      FUNCTION WINDOW(J,MM,WOPT)
*
*  These window functions come from the paper "On the Use of Windows
*  for Harmonic Analysis with the Discrete Foutier Transform", by
*  Fredric J. Harris, IEEE Proc., Vol. 66, No.1, Jan 1978.
```

```
*
*  J = The current time index.
*  MM = The total number of points to be windowed.
*  WOPT = The selected window option.
*
*-----------------------------------------------------------------------
      INTEGER WOPT
      PI2=8.*ATAN(1.)
      M=MM/2
      FACM=M-0.5
      FACP=1./(M+0.5)
      IF(WOPT .EQ. 1) THEN
C      -- PARZEN (TRIANGULAR) WINDOW:
      WINDOW=(1.-ABS(((J-1)-FACM)*FACP))
C
      ELSEIF(WOPT .EQ. 2) THEN
C      -- RECTANGULAR WINDOW: ie. no window
      WINDOW=1.
C
      ELSEIF(WOPT .EQ. 3) THEN
C      -- WELCH (PARABOLIC) WINDOW:
      WINDOW=(1.-(((J-1)-FACM)*FACP)**2)
C
      ELSEIF(WOPT .EQ. 4) THEN
C      -- VAN HANN (SQUARED COSINE) WINDOW:
      WINDOW=0.5-0.5*COS(PI2*(FLOAT(J-1)/FLOAT(MM)))
C
      ELSEIF(WOPT .EQ. 5) THEN
C      -- HAMMING (RAISED COSINE) WINDOW:
      WINDOW=0.538-0.462*COS(PI2*(FLOAT(J-1)/FLOAT(MM)))
C
      ELSEIF(WOPT .EQ. 6) THEN
C      -- BLACKMAN-HARRIS 3 TERM MINIMUM WINDOW:
      WINDOW =.42323-.49755*COS(PI2*(FLOAT(J-1)/FLOAT(MM)))+
     1               .07922*COS(PI2*2.*(FLOAT(J-1)/FLOAT(MM)))
C
      ELSEIF(WOPT .EQ. 7) THEN
C      -- BLACKMAN-HARRIS 4 TERM MINIMUM WINDOW:
      WINDOW =.35875-.48829*COS(PI2*(FLOAT(J-1)/FLOAT(MM)))+
     1               .14128*COS(PI2*2.*(FLOAT(J-1)/FLOAT(MM)))-
     2               .01168*COS(PI2*3.*(FLOAT(J-1)/FLOAT(MM)))
C
      ELSE
         WRITE(*,*) 'INVALID WINDOW SELECTION,  TERMINATE...'
         STOP
      ENDIF
      RETURN
      END
C*********************************************************************
      SUBROUTINE FOUR1(DATA,NN,ISIGN)
C
C   Subroutine from "Numerical Recipes" by Press, Flannery, Teukolsky,
C   and Vetterling, pub 1986, Cambridge Univ. Press.
C
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
```

```
      DIMENSION DATA(*)
      N=2*NN
      J=1
      DO 11 I=1,N,2
        IF(J.GT.I)THEN
          TEMPR=DATA(J)
          TEMPI=DATA(J+1)
          DATA(J)=DATA(I)
          DATA(J+1)=DATA(I+1)
          DATA(I)=TEMPR
          DATA(I+1)=TEMPI
        ENDIF
        M=N/2
1       IF ((M.GE.2).AND.(J.GT.M)) THEN
          J=J-M
          M=M/2
        GO TO 1
        ENDIF
        J=J+M
11    CONTINUE
      MMAX=2
2     IF (N.GT.MMAX) THEN
        ISTEP=2*MMAX
        THETA=6.28318530717959D0/(ISIGN*MMAX)
        WPR=-2.D0*DSIN(0.5D0*THETA)**2
        WPI=DSIN(THETA)
        WR=1.D0
        WI=0.D0
        DO 13 M=1,MMAX,2
          DO 12 I=M,N,ISTEP
            J=I+MMAX
            TEMPR=SNGL(WR)*DATA(J)-SNGL(WI)*DATA(J+1)
            TEMPI=SNGL(WR)*DATA(J+1)+SNGL(WI)*DATA(J)
            DATA(J)=DATA(I)-TEMPR
            DATA(J+1)=DATA(I+1)-TEMPI
            DATA(I)=DATA(I)+TEMPR
            DATA(I+1)=DATA(I+1)+TEMPI
12        CONTINUE
          WTEMP=WR
          WR=WR*WPR-WI*WPI+WR
          WI=WI*WPR+WTEMP*WPI+WI
13      CONTINUE
        MMAX=ISTEP
      GO TO 2
      ENDIF
      RETURN
      END
```

I. USER'S GUIDE FOR PROGRAM ENSPSDAV

1. Input Instructions

- Insure that the executable file for program ENSPSDAV (ENSPSDAV.EXE) and the input data file are in the same subdirectory of a disk drive with sufficient space remaining to write the output file.

- Initiate program ENSPSDAV (type "ENSPSDAV' at the DOS prompt).

- The program will prompt the user for the name of the input data file. Enter the full name of the input data file including the extension.

- The program will prompt the user for the name of the output file which will be created by the program. Enter the desired file name. Do not use the name of an existing file or the program will terminate.

- existing file or the program will terminate.

- The program will prompt the user for the sample rate of the data. Enter the sample rate of the data contained in the input file.

- The program will then read the input data file. Remember to observe the column lengths of the array ENSMBL to insure that the ensemble is being constructed properly.

- Next, the program will prompt the user to choose whether or not to remove the mean of each individual segment prior to computing the spectral estimate of the segment. Enter "Y" for yes (remove the mean) or "N" for no (do not remove the mean). If the data has a large mean value (relative to the magnitude of the oscillations) then it is recommended that the mean be removed.

- The program will prompt the user to select one of the seven tapering window options from Table IV-4. Enter the number of the desired option. (Windows 5 and 6 are good compromises).

- Then the program will prompt the user to select one of three options for segmenting the data. Enter the number of the selected option. Option 12 allows the user to enter the number of samples between the beginning of each

segment. Option 23 allows the user to select the number of segments which span the ensemble (i.e., the number of spectral estimates across the ensemble of pulses). Option 3 allows the user to enter the sample number of the mid-point location of a single segment (i.e., the location in time of a single spectral estimate within the ensemble). All segments are 256 samples wide.

- The program will then compute the spectral estimates of the selected segments and write the output file.

- The program will prompt the user to choose whether to end operation or continue working with this same ensemble of data. If the user chooses to continue, the program prompts for another output file name and the process begins again.

2. Output Description

The output file consists of some heading information and five columns of spectral data. The heading information summarizes the options selected (segmenting option, tapering window option, the mean removal option and the input data file name). Then the location of the first segment spectral estimate is identified. The first column of data is the frequency. The second column contains the magnitude of the power at each listed frequency. The third column presents the relative power in decibels (dB) computed such that the peak power is zero dB. The fourth column presents a running total of the power (magnitude) and the fifth column lists the percent running total power. Then the location of the next segment is identified and the five columns of spectral data are repeated for this next segment, repeating until the spectral results for each ensemble segment have been reported.

The following pages present the source code for program ENSEMBL.

```
$DEBUG
      PROGRAM PSDAVG
************************************************************************
*   Program to perform spectral analysis on short time segments of an
*   ensemble of non-stationary turbulent boundary layer velocity data to
*   obtain the time varying spectral characteristics of the data.
*
*   Input data file is assumed to be in the format of the A/D conversion
*   system, ie. a sequential file with alternating values of velocity (U)
*   and propellor pulse (PROP), starting with velocity.
*
*   The user is given three options on how to segment the ensemble,
*   by specifying the no. of points between the beginning of each segment,
*   the no. of segments across the ensemble, or the center point of a
*   single segment.
*
*   After segmenting per the users choice, and removing the mean (if
*   selected), the program builds array W1 which contains M consecutive
*   time points from a single pulse. This time slice is windowed using the
*   spectral window selected.The program, then computes a PSD estimate on the
*   time slice. This process is repeated for the corresponding time slice
*   in each propeller pulse. Then the PSD of each pulse time slice is
*   averaged to get the final PSD for the segment. This PSD is output
*   and then the entire sequence is repeated on the next segment.
*   The next segment is shifted IWIDE time points, but is always M points
*   long.
*
*   Key variables are defined below:
*
*  ENSMBL = The primary data array which contains up to 50 pulses (columns)
*           each of length up to 2048 points. The pulses are synchronized in
*           time (an ensemble) such that each point in a pulse "lines up"
*           with the others.
*  U = The measured velocity as read from the input file into array ENSMBL.
*  PROP = The propeller pulse timing marker signal. Also read from the input.
*  IPLS = Number of pulses to be read from the input file.
*  LASTI = Number of points in the columns of ENSMBL. (ie. rows in ENSMBL)
*  IWIDE = Number of time points from the beginning of the current segment
*          to the beginning of the next segment.
*  NSEG = Number of time slice segments in the ensemble.
*  NPLS = The actual total number of pulses (columns in ENSMBL) after the
*         input data has been split into an ensemble.
*  ISTART = Starting point number of the segment.
*
*                         PSD Related Variables:
*
*  M = Number of points per segment (MUST BE A POWER OF 2). Also the number
*      of points in each FFT. (Currently fixed at 256 points)
*
*  WINDOW = The current FFT window function value (see function subroutine).
*  WOPT = Window selection parameter:    WOPT            WINDOW
*                                          1             Parzen
*                                          2             Rectangular
*                                          3             Welch (parabolic)
*                                          4             von Hann
```

```
*                                          5              Hamming
*                                          6              3 term B-H
*                                          7              4 term B-H
*
*  See the following reference for details on the window functions:
*  "On the Use of Windows for Harmonic Analysis with the DFT" by
*   Fredric J. Harris, IEEE Proc., Vol. 66, No.1, January 1978
*
*  SMPLRT = Sample rate of the input data.
*  FREQ = Frequency value in HZ of a particular frequenct bin.
*  DFREQ = The width of each frequency bin.
*  W1 = A complex workspace array for the FFT subroutine FOUR1.
*  P = Output PSD array in (ft/sec)**2 per Hz
*  DBP = Output PSD array in decibels per Hz
*  TPWR = The running total power summed from zero frequency to the Jth freq.
*  PCTPWR = Percentage of total power summed from zero frequency to the Jth
*           freq. relative to the total power summed across all freqs.
*
*  Note on dimensions: P is currently dimensioned to M/2+1 (129) frequency
*  bins. W1 is dimensioned to 2*M because it is a complex array. The even
*  elements contain the imaginary part (all zero in this case) and the
*  odd elements contain the real part (the velocity data).
*
*
*                  Programmed by: Donald K. Johnson
*                        August 1988, NPS
*
********************************************************************************
      PARAMETER(MMM=256,MMMO2=128)
      DIMENSION ENSMBL(2048,50),P(MMMO2+1),W1(2*MMM),DBP(MMMO2+1)
      DIMENSION TPWR(MMMO2+1),PCTPWR(MMMO2+1)
      INTEGER I,IPLS,NPLS,J,L,LL,NPTS,IWIDE,WOPT
      REAL FREQ,DFREQ,SMPLRT
      CHARACTER*12 INFILE,OUTFILE
      CHARACTER*1 RM,CONT
C
      PRINT '(/A/)','         Spectral Analysis of Pulse Segments'
      PRINT '(/A\)',' ENTER THE INPUT DATA FILE NAME: '
      READ (*,'(A)') INFILE
C
      PRINT '(/A\)',' ENTER THE OUTPUT DATA FILE NAME: '
      READ (*,'(A)') OUTFILE
C
C     PRINT '(/A\)',' ENTER THE NUMBER OF PROP PULSES YOU WANT: '
C     READ (*,*) IPLS
      IPLS=50
C
      PRINT '(/A\)',' ENTER THE SAMPLE RATE OF THE DATA: '
      READ(*,*) SMPLRT
C
      OPEN(10,FILE=INFILE,STATUS='OLD')
      OPEN(11,FILE=OUTFILE,STATUS='NEW')
C
      LASTI=2048
      J=1
```

```
      I=0
      PRINT'(/A/)',' Reading input file...Watch the no. of points in eac
     1h column!'
*
 10   CONTINUE
***     Read the data into array "ENSMBL", each column is a pulse ***
*       If an EOF is encountered then exit the loop and jump to 20. If no
*       EOF is encountered then jump to 28
        READ (10,*,END=20,ERR=20) U,PROP
        I=I+1
***     But skip the first partial pulse ******
        IF (J .GT. 1) ENSMBL(I,J-1)=U
***     If you have a new pulse, start a new column (pulse trigger currently
*       set to 1 volt.
        IF(I .GE. 20 .AND. PROP .GT. 1.) THEN
            IF (J .EQ. 1) THEN
               WRITE(*,*)'Skipped first',I,' points (not a full pulse).'
            ELSE
               WRITE(*,*)'No. of points in column',J-1,' is',I
            ENDIF
***         Find the shortest column ***
            IF(I .LT. LASTI  .AND.  J .GT. 1) LASTI=I
***         Quit loop if you've read enough pulses ***
            IF (J-1 .GE. IPLS) GO TO 28
            J=J+1
            I=0
        ENDIF
***   Curse me, a go to statement! I never took structured programming. ***
      GO TO 10
 20   CONTINUE
***   since an EOF was encountered, write error message and decrement J to
*     ignore the last partial pulse. This is the normal path.
         WRITE(*,'(/A/)')' End of File encountered - deleting last parti
     1al column....'
         J=J-1
 28   CONTINUE
      XN=FLOAT(J-1)
      NPLS=J-1
      WRITE(*,'(A,I3,A,I6)')' The final number of pulses is',NPLS,' , ea
     1ch of length ',LASTI
C
 29   WRITE(*,'(/,/A\)')' DO YOU WANT TO REMOVE THE MEAN BEFORE COMPUTIN
     1G PSDs? (Y or N): '
      READ(*,'(A)') RM
*
      WRITE(*,'(/A\)')' ENTER FFT "WINDOW" OPTION (1 thru 7): '
      READ(*,*) WOPT
*
*** Menu of options for segmenting ensembles ***
      WRITE(*,'(/A/)')' How do you want to segment the ensemble of pulse
     1s?'
 25   WRITE(*,'(A)')' OPTION                    DESCRIPTION'
      WRITE(*,'(A)')'   1         Select the time width between each PSD'
      WRITE(*,'(A)')'   2         Select the total number of PSDs (segmen
     1ts)'
```

```
      WRITE(*,'(A)')'   3        Select the center time point for a sing
     1le PSD'
      WRITE(*,'(/A\)')' ENTER THE OPTION NUMBER OF CHOICE: '
      READ(*,*) IOPT
*
*** Note that all segment lengths (M) are 256 points wide. Smaller width
*   separations (IWIDE) cause adjacent segments to be overlapped.
*
*** option 1 ***
      IF(IOPT .EQ. 1) THEN
          WRITE(*,'(/A\)')' ENTER THE TIME WIDTH (IN NO. OF POINTS) BETW
     1EEN EACH PSD: '
          READ(*,*) IWIDE
***       this scheme will probably cause you to pick up some zeros after
*         the end of the data - so the last segment may have a discontinuity
          NSEG=(LASTI-M)/IWIDE + 1
          ISTART=1
          M=256
*
*** option 2 ***
      ELSEIF(IOPT .EQ. 2) THEN
          WRITE(*,'(/A\)')' ENTER THE TOTAL NUMBER OF PSDs (SEGMENTS) AC
     1ROSS THE ENSEMBLE: '
          READ(*,*) NSEG
***       you will probably miss a few points at the end of the pulse
          IWIDE=(LASTI-M)/(NSEG-1)
          ISTART=1
          M=256
*
*** option 3 ***
      ELSEIF(IOPT .EQ. 3) THEN
          WRITE(*,'(/A\)')' ENTER THE TIME POINT NUMBER (CENTER POINT) O
     1F A SINGLE 256 POINT TIME SLICE      (at least 128 points from eit
     2her end of the pulse): '
          READ(*,*) ITIME
          ISTART=ITIME-127
          IWIDE=256
          NSEG=1
          M=256
      ELSE
          WRITE(*,'(/A/)')' OOPS! - INVALID OPTION CHOICE, TRY AGAIN'
          GO TO 25
      ENDIF
      IF(IWIDE .GT. M) THEN
         WRITE(*,'(/A/)')' SEGMENT WIDTH .GT. 256 (TOO LARGE).'
         GO TO 25
      ENDIF
*
*** Compute the spectral estimate of an ensemble of pulses in data segments
*   that are M points wide and IWIDE points apart.
*   First initialize some constants.
      ZERO=0.0
      MO2=M/2
      DFREQ=SMPLRT/M
      MM=M+M
```

```
      M44=MM+4
      M43=MM+3
      SUMW=0.
*** sum of square of window function,  used for normalizing psd later ***
      DO 45 J=1,M
         SUMW=SUMW+WINDOW(J,M,WOPT)**2
 45   CONTINUE
*
*** write some heading information to the begining of the output file ***
      WRITE(11,'(A,I3)')' Selected options:  Segment option: ',IOPT
      WRITE(11,'(A,I3)')'                     Window option: ',WOPT
      WRITE(11,'(A,A)')'                    Mean removal: ',RM
      WRITE(11,'(A,A12)')'                    Input file: ',INFILE
*
*** Start the outer loop, the segment loop ****
      DO 50 NN=1,NSEG
      WRITE(*,*)' Computing averaged PSD from segment #',NN
***   initialize spectrum ***
      DO 53 I=1,MO2+1
         P(I)=0.
         TPWR(I)=0.
         PCTPWR(I)=0.
         DBP(I)=0.
 53   CONTINUE
      DEN=0.
*
***     Read M Points from pulse # L (starting with ISTART) and write to
*       ARRAY W1 ***
***     Start the pulse loop ***
        DO 120 L=1,NPLS
***        and the data point (time) loop ***
           DO 100 I=1,M
               II=I+ISTART-1
               I2=I+I
***            put the real values in the odd numbered elements of W1. ***
               W1(I2-1)=ENSMBL(II,L)
***            the imaginary part (even elements) is zero ***
               W1(I2)=0.
 100       CONTINUE
*
***  Now remove the mean (of the segment) if it was selected above ***
           IF(RM .EQ. 'Y' .OR. RM .EQ. 'y') THEN
               SUMRM=0.
               DO 105 I=1,M
                  I2=I+I
                  SUMRM=SUMRM+W1(I2-1)
 105           CONTINUE
               REVMN=SUMRM/M
               DO 106 I=1,M
                  I2=I+I
                  W1(I2-1)=W1(I2-1)-REVMN
 106           CONTINUE
               ENDIF
*
***     Apply the window function to the M data points. Operate only on the
```

```
*       real part of W1 since the imag. part is zero. ****
        DO 160 J=1,M
          J2=J+J
          W=WINDOW(J,M,WOPT)
          W1(J2-1)=W1(J2-1)*W
 160    CONTINUE
*
***     Fourier transform the windowed data, then sum results into previous
*       segment for later averaging ***
        CALL FOUR1(W1,M,1)
***     now sum up the real pos. and neg. frequencies from each segment ***
        P(1)=P(1)+W1(1)**2
        DO 17 J=2,MO2
          J2=J+J
          P(J)=P(J)+W1(J2-1)**2+W1(M43-J2)**2
  17    CONTINUE
        P(MO2+1)=P(MO2+1)+W1(M+1)**2
        DEN=DEN+SUMW
***   now, go get a segment from the next pulse and do it again ***
 120  CONTINUE
*
C *** correct normalization for windowing ***
      DEN=M*4*DEN
C *** normalize the output, ie. compute the average power over all pulses ***
      DO 19 J=1,MO2+1
        P(J)=P(J)/DEN
19    CONTINUE
*
C*** compute the relative PSD in decibels and the running total power ***
*     zero dB is the peak value ***
      PMAX=P(1)
      DO 51 J=1,MO2
         IF(P(J+1) .GT. PMAX) PMAX=P(J+1)
   51 CONTINUE
      SUMPWR=0.
      DO 60 J=1,MO2+1
         IF (P(J) .EQ. 0.) THEN
            WRITE(*,*) 'WARNING: P(J) has a zero value. P(J) set to -999
     1999.0  at a J of',J,'                        ... continuing.'
            DBP(J)=-999999.
         ELSE
            DBP(J)=20.*ALOG10(P(J)/PMAX)
            SUMPWR=SUMPWR+P(J)
            TPWR(J)=SUMPWR
         ENDIF
   60 CONTINUE
*
*** compute the percentage of running total power at freq. J relative to the
*    total power summed over all freqs. ***
      DO 70 J=1,MO2+1
         PCTPWR(J)=100.*TPWR(J)/TPWR(MO2+1)
  70  CONTINUE
***   write out the results for segment # NN ***
       WRITE(11,'(3(A,I6))')' PSD OF TIME SEGMENT',NN,', POINT NO.',IST
     1ART,' THROUGH',ISTART+M-1
```

```
        DO 300 LL=1,MO2+1
           FREQ=LL*DFREQ - DFREQ
           WRITE(11,'(1X,F10.1,F17.3,F10.3,F17.3,F8.2)') FREQ,P(LL),DBP(
     1LL),TPWR(LL),PCTPWR(LL)
  300   CONTINUE
*
***     now get ready for the next segment ***
        ISTART=ISTART+IWIDE
  50  CONTINUE
*
*** All Done ***
*** If desired, go back and run another operation on the same ensemble ***
      WRITE(*,'(/A\)')' DO YOU WISH TO CONTINUE WITH THIS ENSEMBLE? (Y o
     1r N): '
      READ(*,'(A)') CONT
      IF(CONT .EQ. 'Y' .OR. CONT .EQ. 'y') THEN
         CLOSE(11)
         PRINT '(/A\)',' ENTER ANOTHER (NEW) OUTPUT DATA FILE NAME: '
         READ (*,'(A)') OUTFILE
         OPEN(11,FILE=OUTFILE,STATUS='NEW')
         GO TO 29
      ENDIF
      CLOSE(10)
      CLOSE (11)
      END
C*******************************************************************************
      FUNCTION WINDOW(J,MM,WOPT)
*
*  These window functions come from the paper "On the Use of Windows
*  for Harmonic Analysis with the Discrete Foutier Transform", by
*  Fredric J. Harris, IEEE Proc., Vol. 66, No.1, Jan 1978.
*
*
*****************************************************************
      INTEGER WOPT
      PI2=8.*ATAN(1.)
      M=MM/2
      FACM=M-0.5
      FACP=1./(M+0.5)
      IF(WOPT .EQ. 1) THEN
C      -- PARZEN (TRIANGULAR) WINDOW:
      WINDOW=(1.-ABS(((J-1)-FACM)*FACP))
C
      ELSEIF(WOPT .EQ. 2) THEN
C      -- RECTANGULAR WINDOW: ie. no window
      WINDOW=1.
C
      ELSEIF(WOPT .EQ. 3) THEN
C      -- WELCH (PARABOLIC) WINDOW:
      WINDOW=(1.-(((J-1)-FACM)*FACP)**2)
C
      ELSEIF(WOPT .EQ. 4) THEN
C      -- VAN HANN (SQUARED COSINE) WINDOW:
      WINDOW=0.5-0.5*COS(PI2*(FLOAT(J-1)/FLOAT(MM)))
C
```

```
      ELSEIF(WOPT .EQ. 5) THEN
C      -- HAMMING (RAISED COSINE) WINDOW:
      WINDOW=0.538-0.462*COS(PI2*(FLOAT(J-1)/FLOAT(MM)))
C
      ELSEIF(WOPT .EQ. 6) THEN
C      -- BLACKMAN-HARRIS 3 TERM MINIMUM WINDOW:
      WINDOW =.42323-.49755*COS(PI2*(FLOAT(J-1)/FLOAT(MM)))+
     1               .07922*COS(PI2*2.*(FLOAT(J-1)/FLOAT(MM)))
C
      ELSEIF(WOPT .EQ. 7) THEN
C      -- BLACKMAN-HARRIS 4 TERM MINIMUM WINDOW:
      WINDOW =.35875-.48829*COS(PI2*(FLOAT(J-1)/FLOAT(MM)))+
     1               .14128*COS(PI2*2.*(FLOAT(J-1)/FLOAT(MM)))-
     2               .01168*COS(PI2*3.*(FLOAT(J-1)/FLOAT(MM)))
C
      ELSE
         WRITE(*,*) 'INVALID WINDOW SELECTION,  TERMINATE...'
         STOP
      ENDIF
      RETURN
      END
C*********************************************************************
      SUBROUTINE FOUR1(DATA,NN,ISIGN)
*
*  FFT subroutine comes from the book "Numerical Recipes", by Press et.al.
*  Cambridge Univ. Press, 1986.
*
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION DATA(*)
      N=2*NN
      J=1
      DO 11 I=1,N,2
        IF(J.GT.I)THEN
          TEMPR=DATA(J)
          TEMPI=DATA(J+1)
          DATA(J)=DATA(I)
          DATA(J+1)=DATA(I+1)
          DATA(I)=TEMPR
          DATA(I+1)=TEMPI
        ENDIF
        M=N/2
1       IF ((M.GE.2).AND.(J.GT.M)) THEN
          J=J-M
          M=M/2
        GO TO 1
        ENDIF
        J=J+M
11    CONTINUE
      MMAX=2
2     IF (N.GT.MMAX) THEN
        ISTEP=2*MMAX
        THETA=6.28318530717959D0/(ISIGN*MMAX)
        WPR=-2.D0*DSIN(0.5D0*THETA)**2
        WPI=DSIN(THETA)
        WR=1.D0
```

```
        WI=0.D0
        DO 13 M=1,MMAX,2
          DO 12 I=M,N,ISTEP
            J=I+MMAX
            TEMPR=SNGL(WR)*DATA(J)-SNGL(WI)*DATA(J+1)
            TEMPI=SNGL(WR)*DATA(J+1)+SNGL(WI)*DATA(J)
            DATA(J)=DATA(I)-TEMPR
            DATA(J+1)=DATA(I+1)-TEMPI
            DATA(I)=DATA(I)+TEMPR
            DATA(I+1)=DATA(I+1)+TEMPI
12        CONTINUE
          WTEMP=WR
          WR=WR*WPR-WI*WPI+WR
          WI=WI*WPR+WTEMP*WPI+WI
13      CONTINUE
        MMAX=ISTEP
      GO TO 2
      ENDIF
      RETURN
      END
```

J. USER'S GUIDE FOR PROGRAM CORREX

1. Input Instructions

- Insure that the executable file for program CORREX (CORREX.EXE) and the input data file are in the same subdirectory of a disk drive with sufficient space remaining to write the output file. Not that the input file format for program CORREX depends on whether autocorrelation or cross correlation is desired. If autocorrelation is desired, the input file is expected to be a sequential file containing samples of a single variable (e.g., u component velocity). If cross correlation is desired, the input file is expected to be a sequential ASCII file containing alternating samples of two variables (e.g., u component and v component velocity).

- Initiate program CORREX (type "CORREX" at the DOS prompt).

- The program will prompt the user for the name of the input data file. Enter the full name of the input data file including the extension.

- The program will prompt the user for the name of the output file which will be created by the program. Enter the desired file name. Do not use the name of an existing file or the program will terminate.

- The program will prompt the user to select the option for autocorrelation (1) or cross correlation (2). Enter the desired choice.

- The program will prompt the user to enter the number of lags which are desired as a fraction of the total number of data points. For example, if the autocorrelation is desired on a time series containing 1000 points, and the user wishes to compute the autocorrelation for 250 lag points, then 0.25 is entered at the prompt. For truly random data, the autocorrelation usually falls rapidly to near zero such that typical lag values are 0.1 or 0.2

- The program will compute the selected correlation function and write the output file.

2. Output Description

The output file consists of two columns. The first column contains the lag number. The lag number divided by the sample rate yields the value of the lag time, . The second column presents the selected correlation function, corresponding to the lag numbers. The following pages present the source code for program CORREX.

```
$DEBUG
      PROGRAM CORREX
*************************************************************************
*  This program computes an estimate of the autocorrelation function
*  or cross correlation function of a real discrete (stationary) time series
*  using the explicit method. The main program is a driver to set up the
*  input for the call to SUBROUTINE CORMAR. Subroutine CORMAR comes from
*  "Digital Spectral Analysis with Applications" by S. Lawrence Marple,
*  Prentiss-Hall, 1987. The explicit method (see CORMAR) is slower than
*  the more typical FFT methods, but is easier to understand and program.
*  It also has the advantage of not requiring an integer power
*  of 2 number of data points like the FFT versions require.
*
*  For autocorrelation, the input file should be a sequential file containing
*  values of velocity (or whatever single parameter you wish to
*  autocorrelate). The time variable is implied knowing the sample rate.
*
*  For cross correlation, the input file should be a sequential file
*  containing alternating values of the two component velocities at each
*  time sample (or whichever parameters you wish). As above, the time is
*  implied by the sample rate.
*
*  After reading the data into the proper arrays and calibrating, the mean
*  is computed and subtracted from DATA1 and DATA2 to obtain only the
*  fluctuating velocity component u' (and v' for cross correl.). These
*  values are stored back in DATA1 and DATA2. Note, this program only
*  computes correlation for positive lag values. Autocorrelations are
*  even functions, so you don't need the negative lags. For cross correlations
*  simply reverse the input data values so that DATA1 becomes DATA2 and visa
*  versa. This works because  Ruv(-tau) = Rvu(tau).
*
*
*  The key variables are defined below:
*
*  N = Array dimension parameter, maximum number of data points
*  NPTS = Number of velocity input values.
*  DATA1 = First data array (i.e. u')
*  DATA2 = second data array (i.e. v', is set equal to u for autocorrel.)
*  LAGG = Maximum dimension of correlation array.
*  LAG = The number of lags for which to compute the correlation (less than N)
*  R0 = The correlation result at lag=0
*  R = The resulting correlation array for lags 1 through LAG.
*
*              Programmed by: Donald K. Johnson
*                             August 1988, NPS Monterey CA.
*
*************************************************************************
      PARAMETER(N=4096,LAGG=4096)
      DIMENSION DATA1(N),DATA2(N),R(LAGG)
      INTEGER NPTS,LAG
      CHARACTER*12 INFILE,OUTFILE
C
      PRINT '(/A/)',' Computes the auto or cross correlation function
     1using the explicit method. '
C
```

```
        PRINT '(/A\)',' ENTER THE INPUT DATA FILE NAME: '
        READ(*,'(A)') INFILE
        PRINT '(/A\)',' ENTER THE OUTPUT FILE NAME: '
        READ(*,'(A)') OUTFILE
*
        OPEN(9,FILE=INFILE,STATUS='OLD')
        OPEN(10,FILE=OUTFILE,STATUS='NEW')
*
*** choose the input file format ****
 5      PRINT '(/A)',' Do you want to compute autocorrelation <1> or cro
     1ss correlation <2> ?'
        PRINT '(A\)',' ENTER CHOICE: '
        READ(*,*) KOPT
*
        PRINT '(/A\)',' ENTER THE NUMBER OF LAG POINTS (A DECIMAL FRACTI
     1ON OF THE TOTAL NUMBER OF        POINTS - usually around 0.1 or 0.
     22): '
        READ(*,*) FRAC
*
***     currently deactivated
*       PRINT '(/A\)',' ENTER THE CALIBRATION COEFFICIENTS A AND B [wher
*    1e u = A*(data**B)]: '
*       READ(*,*) A,B
*
***   for autocorrelation ***
        IF(KOPT .EQ. 1) THEN
          DO 11 I=1,N
            READ (9,*,END=15,ERR=15) DATA1(I)
*
*** convert the hot wire voltage to velocity in ft./sec. ***
***         currently deactivated
***            DATA1(I)=A*(DATA1(I)**B)
*
            DATA2(I)=DATA1(I)
11        CONTINUE
*
***   for cross correlation ***
        ELSEIF(KOPT .EQ. 2) THEN
          DO 12 I=1,N
            READ(9,*,END=15,ERR=15) DATA1(I),DATA2(I)
*
*** convert the hot wire voltage to velocity in ft./sec. ***
***         currently deactivated
*           DATA1(I)=A*(DATA1(I)**B)
*           DATA2(I)=A*(DATA2(I)**B)
*
12        CONTINUE
        ELSE
          PRINT '(/A/)',' Invalid choice, try again.'
          GO TO 5
        ENDIF
15      CONTINUE
*
*** compute the maximum lag based on the number of data points ***
        NPTS=I
```

```
        LAG=INT(NPTS*FRAC)
*
***   remove the mean    i.e. compute u' (and v') ***
           SUMD1=0.
           SUMD2=0.
        DO 100 I=1,NPTS
           SUMD1=SUMD1+DATA1(I)
           SUMD2=SUMD2+DATA2(I)
 100    CONTINUE
        AVG1=SUMD1/NPTS
        AVG2=SUMD2/NPTS
        DO 200 I=1,NPTS
           DATA1(I)=DATA1(I)-AVG1
           DATA2(I)=DATA2(I)-AVG2
 200    CONTINUE
*
***     compute the unbiased correlation estimate  ****
*
        PRINT '(/A\)',' Computing correlation ...'
        CALL CORMAR (NPTS,LAG,0,DATA1,DATA2,R0,R)
*
***     Write the output. Note, this program only computes correlations
*       for positive lags.
        IZERO=0
            WRITE(10,'(1X,I5,3X,F15.3)') IZERO,R0
        DO 13 I=1,LAG
            WRITE(10,'(1X,I5,3X,F15.3)') I,R(I)
13      CONTINUE
        CLOSE(9)
        CLOSE(10)
        END
*********************************************************************
      SUBROUTINE CORMAR (N,LAG,MODE,X,Y,R0,R)
C
C   Subroutine from S.L. Marple's "Digital Spectral Analysis with
C   Applications", Prentiss-Hall, 1987
C
C   Modified for real valued inputs by Donald Johnson, Aug. 1988, Naval
C   Postgraduate School, Monterey, CA.
C
C   This program computes either the unbiased or biased real correlation
C   estimates between real data sample arrays X and Y.  If X=Y, then the
C   autocorrelation is computed.
C
C   Input Parameters:
C
C     N    - Number of data samples in arrays X and Y (integer)
C     LAG  - Number of correlation lags to compute [ lags from  0  to  LAG
C              are computed and stored in R0 and R(1) to R(LAG) ] (integer)
C     MODE - Set to 0 for unbiased correlation estimates; otherwise, biased
C              correlation estimates are computed (integer)
C     X    - Array of real data samples X(1) through X(N)
C     Y    - Array of real data samples Y(1) through Y(N)
C
C   Output Parameters:
```

```
C
C     R0    - Real correlation estimate for lag 0
C     R     - Array of real correlation estimates for lags 1 to LAG
C
C   Notes:
C
C     External arrays X,Y must be dimensioned .GE. N and array R must be
C     dimensioned .GE. LAG in the calling program.
C
      REAL X(1),Y(1),R(1),R0,SUM
      INTEGER N,LAG,K,MODE,NK
      DO 30 K=0,LAG
        NK=N-K
        SUM=0.
        DO 10 J=1,NK
10        SUM=SUM+X(J+K)*Y(J)
        IF (K .NE. 0)  GO TO 20
        R0=SUM/FLOAT(N)
        GO TO 30
20      IF (MODE .EQ. 0)  R(K)=SUM/FLOAT(N-K)
        IF (MODE .NE. 0)  R(K)=SUM/FLOAT(N)
30      CONTINUE
      RETURN
      END
```

K. USER'S GUIDE FOR PROGRAM CORFFT

1. Input Instructions

- Insure that the executable file for program CORFFT (CORFFT.EXE) and the input data file are in the same subdirectory of a disk drive with sufficient space remaining to write the output file. Not that the input file format for program CORFFT depends on whether autocorrelation or cross correlation is desired. If autocorrelation is desired, the input file is expected to be a sequential file containing samples of a single variable (e.g., u component velocity). If cross correlation is desired, the input file is expected to be a sequential ASCII file containing alternating samples of two variables (e.g., u component and v component velocity).

- Initiate program CORFFT (type "CORFFT' at the DOS prompt).

- The program will prompt the user for the name of the input data file. Enter the full name of the input data file including the extension.

- The program will prompt the user for the name of the output file which will be created by the program. Enter the desired file name. Do not use the name of an existing file of the program will terminate.

- The program will prompt the user to select the option for autocorrealtion (1) or cross correlation (2). Enter the number of the desired choice.

- The program will prompt the user to enter the number of lags which are desired as a fraction of the total number of data points. For example, if the autocorrelation is desired on a time series containing 1000 points, and the user wishes to compute the autocorrelation for 250 lag points, then 0.25 is entered at the prompt. For truly random data, the autocorrelation usually falls rapidly to near zero such that typical lag values are 0.1 or 0.2.

- The program will compute the selected correlation function and write the output file.

2. Output Description

The output file consists of two columns. The first column contains the lag number. The lag number divided by the sample rate yields the value of the lag time, . The second column presents the selected correlation function, corresponding to the lag numbers. The following pages present the source code for program CORFFT.

```
$DEBUG
      PROGRAM CORFFT
*************************************************************************
*  This program computes an estimate of the autocorrelation function
*  or cross correlation function of a real discrete (stationary) time
*  series using the Fourier transform method. The main program is a driver
*  to set up the input for the call to SUBROUTINE CORREL. Subroutine CORREL,
*  TWOFFT, REALFT AND FOUR1 come from "Numerical Recipes" by Press et. al.,
*  Cambridge, 1986.
*
*  For autocorrelation, the input file should be a sequential file containing
*  values of velocity (or whatever parameter you wish to autocorrelate).
*  The time variable is implied knowing the sample rate.
*
*  For cross correlation, the input file should be a sequential file
*  containing alternating values of the two component velocities at each
*  time sample (or whichever parameters you wish). As above, the time is
*  implied by the sample rate.
*
*  After reading the data into the proper arrays and calibrating, the mean
*  is computed and subtracted from DATA1 and DATA2 to obtain only the
*  fluctuating velocity component u' (and v' for cross correl.). These
*  values are stored back in DATA1 and DATA2.
*
*  The Fourier transform method of computing correlation results in a
*  circular correlation. Two convert this to linear correlation, the elements
*  of array ANS are divided by (NPTS-I) where I is the current lag value. I
*  varies from 0 to LAG. This program is only outputting correlation values
*  at positive lags; ANS(1) for zero lag to ANS(N/2) for lag=LAG.
*  Autocorrelations of real time series are even functions so yopu don't
*  need the negative lags. For cross correlations you may want to see the
*  negative lags, so just modify the output WRITE statements or reverse the
*  input data vectors. Reversing the input time series will give you the
*  negative lag values because  Ruv(-tau) = Rvu(tau). Increasingly
*  negative lags are stored in wraparound mode from ANS(N) for lag -1
*  to ANS(N-LAG+1) for -LAG.
*
*
*  The key variables are defined below:
*
*  N = Array dimension parameter, maximum number of data points
*  NPTS = Number of velocity input values.
*  DATA1 = First data array (i.e. u')
*  DATA2 = second data array (i.e. v', is set equal to u for autocorrel.)
*  INT2 = The size of DATA1 and DATA2 after zero padding to the smallest
*         integer power of 2 greater than NPTS+LAG. (The size of the FFT.)
*  LAG = The number of lags for which to compute the correlation (less than N)
*  ANS = The circular correlation result for lags 0 through LAG.
*
*              Programmed by: Donald K. Johnson
*                             August 1988, NPS Monterey CA.
*
*************************************************************************
      PARAMETER(N=4096,N2=2*N)
      DIMENSION DATA1(N),DATA2(N),ANS(N2)
```

```
      INTEGER NPTS,LAG
      CHARACTER*12 INFILE,OUTFILE
C
      PRINT '(/A/)',' Computes the auto or cross correlation function
     1using Fourier transform methods.'
C
      PRINT '(/A\)',' ENTER THE INPUT DATA FILE NAME: '
      READ(*,'(A)') INFILE
      PRINT '(/A\)',' ENTER THE OUTPUT FILE NAME: '
      READ(*,'(A)') OUTFILE
*
      OPEN(9,FILE=INFILE,STATUS='OLD')
      OPEN(10,FILE=OUTFILE,STATUS='NEW')
*
***   choose the input file format ****
 5    PRINT '(/A)',' Do you want to compute autocorrelation <1> or cro
     1ss correlation <2> ?'
      PRINT '(A\)',' ENTER CHOICE: '
      READ(*,*) KOPT
*
      PRINT '(/A\)',' ENTER THE NUMBER OF LAG POINTS (A DECIMAL FRACTI
     1ON OF THE TOTAL NUMBER OF        POINTS - usually around 0.1 or 0.
     22): '
      READ(*,*) FRAC
*
***   currently deactivated
*     PRINT '(/A\)',' ENTER THE CALIBRATION COEFFICIENTS A AND B [wher
*    1e u = A*(data**B)]: '
*     READ(*,*) A,B
*
***   for autocorrelation ***
      IF(KOPT .EQ. 1) THEN
        DO 11 I=1,N
          READ (9,*,END=15,ERR=15) DATA1(I)
*
***       convert the hot wire voltage to velocity in ft./sec. ***
***       currently deactivated
*         DATA1(I)=A*(DATA1(I)**B)
*
          DATA2(I)=DATA1(I)
11      CONTINUE
*
***   for cross correlation ***
      ELSEIF(KOPT .EQ. 2) THEN
        DO 12 I=1,N
          READ(9,*,END=15,ERR=15) DATA1(I),DATA2(I)
*
***       convert the hot wire voltage to velocity in ft./sec. ***
***       currently deactivated
*         DATA1(I)=A*(DATA1(I)**B)
*         DATA2(I)=A*(DATA2(I)**B)
*
12      CONTINUE
      ELSE
        PRINT '(/A/)',' Invalid choice, try again.'
```

```
          GO TO 5
        ENDIF
15      CONTINUE
*
***     compute the maximum lag based on the number of data points ***
        NPTS=I
        LAG=INT(NPTS*FRAC)
*
***     zero pad the data arrays with at least LAG zeros such that the data
*       array is an integer power of two. Max array size is currently 4096.
        DO 25 J=1,12
           INT2=2**J
***        find the smallest power of 2 array size bigger than NPTS+LAG ***
           IF(INT2 .GT. NPTS+LAG) GO TO 27
 25     CONTINUE
 27     CONTINUE
***     zero pad from NPTS to INT2
        DO 30 I=NPTS+1,INT2
           DATA1(I)=0.
           DATA2(I)=0.
 30     CONTINUE
*
***     remove the mean    i.e. compute u' (and v') ***
        SUMD1=0.
        SUMD2=0.
        DO 100 I=1,NPTS
           SUMD1=SUMD1+DATA1(I)
           SUMD2=SUMD2+DATA2(I)
 100    CONTINUE
        AVG1=SUMD1/NPTS
        AVG2=SUMD2/NPTS
        DO 200 I=1,NPTS
           DATA1(I)=DATA1(I)-AVG1
           DATA2(I)=DATA2(I)-AVG2
 200    CONTINUE
*
***     compute the unbiased correlation estimate  ****
        PRINT '(/A\)',' Computing correlation with FFT method...'
        CALL CORREL (DATA1,DATA2,INT2,ANS)
*
***     Write the output.  Note: ANS is divided by NPTS-I to convert the
*       circular correlation to linear correlation. Also, only the positive
*       lags are output. To get the increasingly negative lags, run a DO
*       loop backwards from NPTS (lag of -1) to NPTS-LAG+1  (lag of -LAG).
*
        DO 13 I=0,LAG-1
            WRITE(10,'(1X,I5,3X,F15.3)') I,ANS(I+1)/(NPTS-I)
13      CONTINUE
        CLOSE(9)
        CLOSE(10)
        END
*************************************************************************
      SUBROUTINE CORREL(DATA1,DATA2,N,ANS)
*
*  From "Numerical Recipes" by Press et.al., Cambridge, 1986.
```

```
*
*  Computes the correlation of two real data sets DATA1 and DATA2, each of
*  length N (including zero padding). N must be an integer power of 2. The
*  answer is returned as the first N points in ANS stored in wraparound
*  order. (Increasingly negative lags are in ANS(N) down to ANS(N/2+1),
*  while increasingly positive lags are in ANS(1) up to ANS(N/2) ). ANS must
*  be at least length 2*N in the calling program since it is also used as
*  workspace. Sign convention: if DATA1 lags DATA2 then ANS will show a peak
*  at positive lags.
*
***    maximum FFT size ***
       PARAMETER(NMAX=8192)
       DIMENSION DATA1(N),DATA2(N)
       COMPLEX FFT(NMAX),ANS(N)
***    Fourier transform both data vectors at once ***
       CALL TWOFFT(DATA1,DATA2,FFT,ANS,N)
***    normalize for inverse FFT ***
       NO2=FLOAT(N)/2.0
       DO 11 I=1,N/2+1
***      multiply to find the FFT of their correlation ***
         ANS(I)=FFT(I)*CONJG(ANS(I))/NO2
11     CONTINUE
       ANS(1)=CMPLX(REAL(ANS(1)),REAL(ANS(N/2+1)))
***    inverse transform to obtain circular correlation ***
       CALL REALFT(ANS,N/2,-1)
       RETURN
       END
C*********************************************************************
       SUBROUTINE TWOFFT(DATA1,DATA2,FFT1,FFT2,N)
*
*  From "Numerical Recipes" by Press et.al., Cambridge, 1986.
*
       DIMENSION DATA1(N),DATA2(N)
       COMPLEX FFT1(N),FFT2(N),H1,H2,C1,C2
       C1=CMPLX(0.5,0.0)
       C2=CMPLX(0.0,-0.5)
       DO 11 J=1,N
         FFT1(J)=CMPLX(DATA1(J),DATA2(J))
11     CONTINUE
       CALL FOUR1(FFT1,N,1)
       FFT2(1)=CMPLX(AIMAG(FFT1(1)),0.0)
       FFT1(1)=CMPLX(REAL(FFT1(1)),0.0)
       N2=N+2
       DO 12 J=2,N/2+1
         H1=C1*(FFT1(J)+CONJG(FFT1(N2-J)))
         H2=C2*(FFT1(J)-CONJG(FFT1(N2-J)))
         FFT1(J)=H1
         FFT1(N2-J)=CONJG(H1)
         FFT2(J)=H2
         FFT2(N2-J)=CONJG(H2)
12     CONTINUE
       RETURN
       END
C*********************************************************************
       SUBROUTINE REALFT(DATA,N,ISIGN)
```

```
*
*  From "Numerical Recipes" by Press et.al., Cambridge, 1986.
*
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION DATA(*)
      THETA=6.28318530717959D0/2.0D0/DBLE(N)
      C1=0.5
      IF (ISIGN.EQ.1) THEN
        C2=-0.5
        CALL FOUR1(DATA,N,+1)
      ELSE
        C2=0.5
        THETA=-THETA
      ENDIF
      WPR=-2.0D0*DSIN(0.5D0*THETA)**2
      WPI=DSIN(THETA)
      WR=1.0D0+WPR
      WI=WPI
      N2P3=2*N+3
      DO 11 I=2,N/2+1
        I1=2*I-1
        I2=I1+1
        I3=N2P3-I2
        I4=I3+1
        WRS=SNGL(WR)
        WIS=SNGL(WI)
        H1R=C1*(DATA(I1)+DATA(I3))
        H1I=C1*(DATA(I2)-DATA(I4))
        H2R=-C2*(DATA(I2)+DATA(I4))
        H2I=C2*(DATA(I1)-DATA(I3))
        DATA(I1)=H1R+WRS*H2R-WIS*H2I
        DATA(I2)=H1I+WRS*H2I+WIS*H2R
        DATA(I3)=H1R-WRS*H2R+WIS*H2I
        DATA(I4)=-H1I+WRS*H2I+WIS*H2R
        WTEMP=WR
        WR=WR*WPR-WI*WPI+WR
        WI=WI*WPR+WTEMP*WPI+WI
11    CONTINUE
      IF (ISIGN.EQ.1) THEN
        H1R=DATA(1)
        DATA(1)=H1R+DATA(2)
        DATA(2)=H1R-DATA(2)
      ELSE
        H1R=DATA(1)
        DATA(1)=C1*(H1R+DATA(2))
        DATA(2)=C1*(H1R-DATA(2))
        CALL FOUR1(DATA,N,-1)
      ENDIF
      RETURN
      END
C*********************************************************************
      SUBROUTINE FOUR1(DATA,NN,ISIGN)
*
*  From "Numerical Recipes" by Press et.al., Cambridge, 1986.
*
```

```
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION DATA(*)
      N=2*NN
      J=1
      DO 11 I=1,N,2
        IF(J.GT.I)THEN
          TEMPR=DATA(J)
          TEMPI=DATA(J+1)
          DATA(J)=DATA(I)
          DATA(J+1)=DATA(I+1)
          DATA(I)=TEMPR
          DATA(I+1)=TEMPI
        ENDIF
        M=N/2
1       IF ((M.GE.2).AND.(J.GT.M)) THEN
          J=J-M
          M=M/2
        GO TO 1
        ENDIF
        J=J+M
11    CONTINUE
      MMAX=2
2     IF (N.GT.MMAX) THEN
        ISTEP=2*MMAX
        THETA=6.28318530717959D0/(ISIGN*MMAX)
        WPR=-2.D0*DSIN(0.5D0*THETA)**2
        WPI=DSIN(THETA)
        WR=1.D0
        WI=0.D0
        DO 13 M=1,MMAX,2
          DO 12 I=M,N,ISTEP
            J=I+MMAX
            TEMPR=SNGL(WR)*DATA(J)-SNGL(WI)*DATA(J+1)
            TEMPI=SNGL(WR)*DATA(J+1)+SNGL(WI)*DATA(J)
            DATA(J)=DATA(I)-TEMPR
            DATA(J+1)=DATA(I+1)-TEMPI
            DATA(I)=DATA(I)+TEMPR
            DATA(I+1)=DATA(I+1)+TEMPI
12        CONTINUE
          WTEMP=WR
          WR=WR*WPR-WI*WPI+WR
          WI=WI*WPR+WTEMP*WPI+WI
13      CONTINUE
        MMAX=ISTEP
      GO TO 2
      ENDIF
      RETURN
      END
```

L. SOURCE CODE FOR PROGRAM MEMPSD

The following pages present the source code for program MEMPSD. This program is included for the sake of completeness but without documentation.

```
      PROGRAM MEMPSD
*
*     Driver for routine MEMCOF and EVLMEM - Maximum Entropy Method
*
      PARAMETER(N=16384,M=100,NFDT=512)
      DIMENSION DATA(N),COF(M),WK1(N),WK2(N),WKM(M)
* identify input and output files
      OPEN(5,FILE='RECORD1.DAT',STATUS='OLD')
      OPEN(7,FILE='PSDOUT1.DAT',STATUS='NEW')
* read input file
      READ(5,*) (DATA(I),PROP,I=1,N)
      CLOSE(5)
* compute coefficients of spectral estimate
      CALL MEMCOF(DATA,N,M,PM,COF,WK1,WK2,WKM)
* write output
      WRITE(7,*) 'Power spectrum estimate of data in RECORD1.DAT'
      WRITE(7,'(1X,T6,A,T25,A,T40,A)') 'f*delta','freq','power'
      SMPLRT=30000.
      DO 11 I=0,NFDT
            FDT=0.5*I/NFDT
            FREQ=FDT*SMPLRT
* function EVLMEM evaluates the spectral density function using the
* coefficients computed with MEMCOF.
            WRITE(7,'(3(2X,F14.4))') FDT,FREQ,EVLMEM(FDT,COF,M,PM)
11    CONTINUE
      END
C*********************************************************************
      SUBROUTINE MEMCOF(DATA,N,M,PM,COF,WK1,WK2,WKM)
*
*   From the book "Numerical Recipes" by Press, et al., Cambridge University
*   press, 1986. See this book for documentation.
*
*
      DIMENSION DATA(N),COF(M),WK1(N),WK2(N),WKM(M)
      P=0.
      DO 11 J=1,N
        P=P+DATA(J)**2
11    CONTINUE
      PM=P/N
      WK1(1)=DATA(1)
      WK2(N-1)=DATA(N)
      DO 12 J=2,N-1
        WK1(J)=DATA(J)
        WK2(J-1)=DATA(J)
12    CONTINUE
      DO 17 K=1,M
        PNEUM=0.
        DENOM=0.
        DO 13 J=1,N-K
          PNEUM=PNEUM+WK1(J)*WK2(J)
          DENOM=DENOM+WK1(J)**2+WK2(J)**2
13      CONTINUE
        COF(K)=2.*PNEUM/DENOM
        PM=PM*(1.-COF(K)**2)
        IF(K.NE.1)THEN
```

```
          DO 14 I=1,K-1
            COF(I)=WKM(I)-COF(K)*WKM(K-I)
14        CONTINUE
        ENDIF
        IF(K.EQ.M)RETURN
        DO 15 I=1,K
          WKM(I)=COF(I)
15      CONTINUE
        DO 16 J=1,N-K-1
          WK1(J)=WK1(J)-WKM(K)*WK2(J)
          WK2(J)=WK2(J+1)-WKM(K)*WK1(J+1)
16      CONTINUE
17    CONTINUE
      PAUSE 'never get here'
      END
C*************************************************************
      FUNCTION EVLMEM(FDT,COF,M,PM)
*
*   From the book "Numerical Recipes" by Press, et al., Cambridge University
*   press, 1986. See this book for documentation.
*
*
      DIMENSION COF(M)
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      THETA=6.28318530717959D0*FDT
      WPR=DCOS(THETA)
      WPI=DSIN(THETA)
      WR=1.D0
      WI=0.D0
      SUMR=1.
      SUMI=0.
      DO 11 I=1,M
        WTEMP=WR
        WR=WR*WPR-WI*WPI
        WI=WI*WPR+WTEMP*WPI
        SUMR=SUMR-COF(I)*SNGL(WR)
        SUMI=SUMI-COF(I)*SNGL(WI)
11    CONTINUE
      EVLMEM=PM/(SUMR**2+SUMI**2)
      RETURN
      END
```

LIST OF REFERENCES

1. Holmes, B.J., Obara, C.J., and Yip, L.P., Natural Laminar Flow Experiments on Modern Airplane Surfaces, NASA TP 2256, 1984.

2. Howard, R.M., An Investigation of the Effects of the Propeller Slipstream on a Wing Boundary Layer, Ph.D. Dissertation, Texas A & M University, College Station, Texas, May 1987.

3. Bradshaw, P., An Introduction to Turbulence and Its Measurement, Pergamon Press Inc., 1971.

4. Cebeci, T., and Smith, A.M.O., Analysis of Turbulent Boundary Layers, Academic Press Inc., 1974.

5. Frost, W., and Moulden, T.H., Handbook of Turbulence: Volume 1--Fundamentals and Applications, Plenum Press, 1977.

6. Bendat, J.S., and Piersol, A.G., Random Data Analysis and Measurement Procedures, 2nd ed., John Wiley & Sons, Inc., 1986.

7. Brigham, E.O., The Fast Fourier Transform, Prentice-Hall, Inc., 1974.

8. Strum, R.D., and Kirk, D.E., First Principles of Discrete Systems and Digital Signal Processing, Addison-Wesley Publishing Co., 1988.

9. Hamming, R.W., Digital Filters, 2nd ed., Prentice-Hall, Inc., 1983.

10. Marple, S.L., Digital Spectral Analysis with Applications, Prentice-Hall, Inc., 1987.

11. Otnes, R.K., and Enochson,L., Digital Time Series Analysis, John Wiley & Sons, Inc., 1972.

12. Press, W.H., et al., Numerical Recipes: The Art of Scientific Computing, Cambridge University Press, 1986.

13. Harris, F.J., On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform, Proceedings of the IEEE, Vol. 66, No. 1, January 1978.

14. Landrum, D.B., and Macha, J.M., Influence of a Heated Leading Edge on Boundary Layer Growth, Stability and Transition, paper presented at the AIAA 19th Fluid Dynamics, Plasma Dynamics and Lasers Conference, Honolulu, Hawaii, 8-10 June, 1987.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Chairman Department of Aeronautics, Code 67 Naval Postgraduate School Monterey, CA 93943-5000	1
4.	Chairman Department of Mathematics, Code 53 Naval Postgraduate School Monterey, CA 93943-5000	1
5.	Commander Naval Air Systems Command Washington, D.C. 20360	1
6.	Office of Naval Research ONR 100 800 N. Quincy Street Arlington, VA 22217	1
7.	NASA Langley Research Center MS/185 Technical Library Hampton, VA 23665	1
8.	NASA Ames Research Center Technical Library Moffett Field, CA 94035	1
9.	Technical Director 6520 Test Group/EN Edwards AFB, CA 93523	1
10.	Flight Dynamics Division 6520 Test Group/ ENF Edwards AFB, CA 93523	1
11.	Technical Library 6520 Test Group/ENXL Edwards AFB, CA 93523	1

12. Technical Director 1
Air Force Flight Test Center/CA
Edwards AFB, CA 93523

13. Professor R.M. Howard 7
Department of Aeronautics, Code 67Ho
Naval Postgraduate School
Monterey, CA 93943-5000

14. Professor Paul Ilacqua 2
1470 Calabazas Boulevard
Santa Clara, CA 95051

15. Professor D. Danielson 1
Department of Mathematics, Code 53DD
Naval Postgraduate School
Monterey, CA 93943-5000

16. Professor R. Hippenstiel 1
Department of Electrical and Computer Engineering,
Code 62Hi
Naval Postgraduate School
Monterey, CA 93943-5000

17. Dr. Bruce J. Holmes 1
Flight Applications Branch
MS 247
NASA Langley Research Center
Hampton, VA 23665

18. Donald K. Johnson 2
6520 Test Group/ENF
Edwards AFB, CA 93523

19. Professor C. Therrien 2
Department of Electrical and Computer Engineering,
Code 62Ti
Naval Postgraduate School
Monterey, CA 93943-5000

Thesis
J5771 Johnson
c.1 A data analysis system
for unsteady turbulence
measurements.